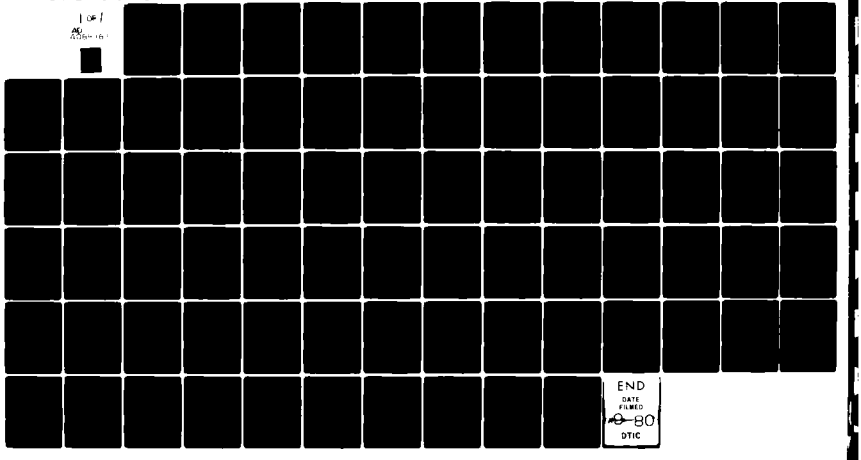
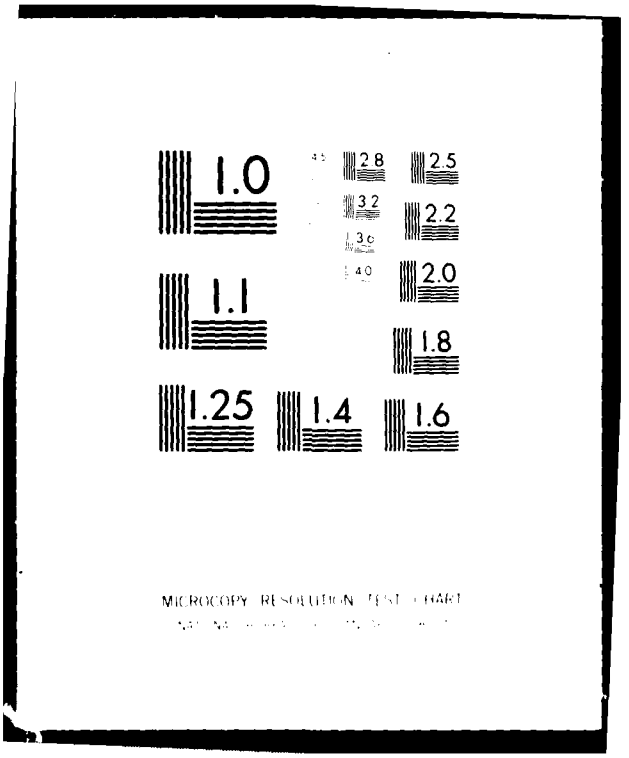
AD-A088 167 ARMY SATELLITE COMMUNICATIONS AGENCY FORT MONMOUTH NJ F/G 9/2
CAT - A DISK CATALOGING SYSTEM FOR THE CP/M OPERATING SYSTEM.(U)
AUG 80 R L CONN

UNCLASSIFIED NL




1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6
MICROCOPY RESOLUTION TEST CHART

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD A088167	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) CAT — A Disk Cataloging System for the CP/M Operating System	5. TYPE OF REPORT & PERIOD COVERED Final 1/80 to 5/80
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Richard L. Conn	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS US Army Satellite Communications Agency USA CORADCOM, Attn: DRCPM-SC-4G Ft Monmouth, NJ 07703	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Elt: 6.11.01.A Proj: 1L1 6110I A91A Task: 33 Work Unit: 131
11. CONTROLLING OFFICE NAME AND ADDRESS -Same as 9-	12. REPORT DATE 4 August 1980
	13. NUMBER OF PAGES 75
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Final rpt Jan-May 80	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Distribution Unlimited

18. SUPPLEMENTARY NOTES

Source Listings to the CAT Program (CAT.ASM) are included as part of this report.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Z80 microprocessor, 8080 microprocessor, CP/M, floppy disk, catalog

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Disk Cataloging System (CAT) described by this document is a utility program which runs on the CP/M operating system, Version 1.4. This system consists of the CAT program itself as a .COM file, the NAMEDISK program as a .COM file, and one or more catalog files (named catname.CAT). It provides an automated method of cataloging the contents of a group of disks and quickly scanning the catalog file so created to determine the disks specified files reside on. This system permits the user to keep a number of distinct

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

catalogs on the same disk and modify the contents of and make queries of each without affecting the others.

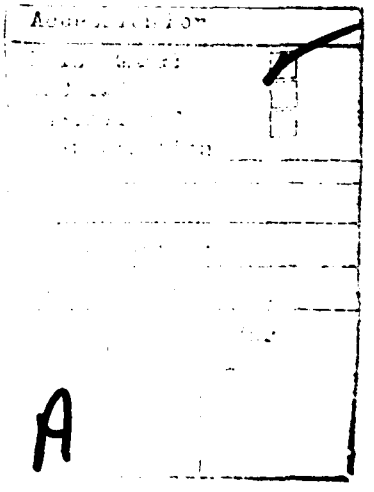
Wild card specifications are permitted, thereby giving the user to selectively catalog files from other disks and selectively scan a catalog file for entries matching a general specification. For example, by specifying the command string "CAT *.ASM *.A??" the user instruction causes CAT to search the current default catalog file for all assembly language source files on all disks in group A (*.A?? specifies any disk name in group A with any disk number). Updating of a catalog can be done in a similar fashion.

CAT — A Disk Cataloging System for the CP/M Operating System

by

Richard Conn

4 August 1980



CONTENTS

The User's Manual for CAT

CHAPTER 1

Introduction

CAT (short for CATALOG) is a program whose function is to maintain and selectively display elements of a catalog of a group of disks. This catalog contains the names of all files on these disks except for those specifically omitted by the user and the names of the disks themselves. CAT is designed to give the user the ability to quickly answer the question: What files do I have and where are they?

This is a relatively simple question, but, if you are a user of CP/M who has more than 10 disks with files on them, you may not find this to be an easy question to answer quickly. Manual systems of keeping track of this information are sometimes quite good; however, as the number of disks grows and the amount of information to keep track of grows correspondingly, the burdon of housekeeping becomes cumbersome. CAT is a program which is designed to relieve this burdon.

CAT performs three basic functions -- CATALOG, UPDATE, and STATUS. The CATALOG function reports on the contents of a CATALOG File. The user may specify the files he wants to search for and the disks he wants to search on, and the CATALOG Function will scan a CATALOG File and list all files matching his specification and all disks said files reside on. The UPDATE function updates the contents of a CATALOG File by examining the disk directory of the disk to be updated, selecting those files specified by the user, and changing the contents of the CATALOG File. Thirdly, the STATUS function reports on the current environment of the CAT program, including such information as the default function and the name of the CATALOG File.

CAT supports many useful and unique features, including:

-- Wild card specification of file names and disk names

-- Negative selection (i.e., specifying file names NOT to be selected)

-- Flexibility in setting and changing the default conditions of CAT itself

-- The ability to support more than one CATALOG File

-- User-selectable Printer output from the CATALOG Function and

Console output from the UPDATE Function

CAT, then, is an attempt to answer a simple question, and it is felt that as the user becomes increasingly more familiar with the usage of this program, he will find this simple question quite easy to answer. This manual is a user's guide to the use of CAT. The first chapters describe CAT and how to use it, and the latter chapters contain error information and a sample terminal session containing displays of output from CAT exactly (or nearly so) as it would appear on the user's console.

CHAPTER 2

The CATALOG File and NAMEDISK

The CATALOG File is the file containing the information for CAT to build upon and retrieve and display to the user. It is a file named

filename.CAT

and it is divided into two parts --

— A list of files to be excluded from the CATALOG File for every disk

— The names of the files and disks in the CATALOG File

The CATALOG File is a CP/M-standard ASCII text file and it may be edited by a CP/M compatable editor. The first character of a CATALOG File MUST be a '('. The list of files to be excluded from the CATALOG File is enclosed by '(' and ')'. This is followed by the names of the files to be excluded (one per line) and a closing ')' as the last character of a line. Examples of a list of excluded files are --

Example 1	Example 2	Example 3	Example 4
(	(	(XDIR.COM	(XDIR.COM
)	XDIR.COM	PIP.COM)	PIP.COM
	PIP.COM		)
	)		

In Example 1, no files are excluded; in Examples 2-4, the files XDIR.COM and PIP.COM are excluded. It is common practice to exclude those files which appear on all (or almost all) of the disks since such information leads to a large number of entries for these files being made in the CATALOG File.

The rest of a CATALOG File contains the names of the files and disks in the CATALOG. This list is in alphabetical order by file name, file type, disk name, and disk type. The file names/types are separated from

the disk names/types by a comma. For example,

Example 1

```
(
XDIR.COM
PIP.COM
)
-SOURCE.A00,SOURCE.A00
-SOURCE.A01,SOURCE.A01
FILE1.ASM,SOURCE.A00
FILE1.ASM,SOURCE.A01
FILE1.COM,SOURCE.A00
MYFILE.ASM,SOURCE.A01
MYFILE.COM,SOURCE.A01
TEST.ASM,SOURCE.A00
```

Example 2

```
(XDIR.COM
PIP.COM)
-SOURCE.A00,SOURCE.A00
-SOURCE.A01,SOURCE.A01
FILE1.ASM,SOURCE.A00
FILE1.ASM,SOURCE.A01
FILE1.COM,SOURCE.A00
MYFILE.ASM,SOURCE.A01
MYFILE.COM,SOURCE.A01
TEST.ASM,SOURCE.A00
```

In the above example, both files are valid CATALOG Files. Note that in both cases, the files XDIR.COM and PIP.COM are excluded from the entries and the entries are in alphabetical order by ASCII collating sequence.

The first entries of any CATALOG File will always be the excluded file list, and an empty CATALOG File will contain just this information. If the CATALOG File is not empty, the next group of entries will be the names of the disks in the catalog preceded by '-' signs. This situation occurs because all disks are named under CAT, and these names take the form of an empty file whose name is '-dskname.typ'. To allow the user to easily name his disks, the program NAMEDISK is used in conjunction with the CAT program. It is executed by mounting a disk to be named in drive B: and typing NAMEDISK at the console (assuming that a disk is mounted in drive A: containing the CAT and NAMEDISK programs). The general form for invoking NAMEDISK is 'NAMEDISK x:', and if 'x:' is specified, then that particular disk will be named (like, 'NAMEDISK A:' names the disk in drive A:), and the disk in drive B: will be named if 'x:' is not specified. When NAMEDISK prompts for a disk name and type, the user should type 'dskname.typ', and the file '-dskname.typ' will be created, thereby naming the disk. The normal CP/M intra-line editing commands (^U, <DEL>, ^R, etc.) are supported by NAMEDISK.

The remaining entries in a CATALOG File are the names of the files and disks in alphabetical order.

CATALOG Files must be initialized by the user. "Empty" CATALOG Files contain just the names of the excluded files surrounded by '(' and ')'. Once initialized, CAT can be used to make all other entries into the files.

CHAPTER 3

The CATALOG, UPDATE, and STATUS Functions

As mentioned earlier, the CAT program performs three basic functions -- CATALOG, UPDATE, and STATUS. The CATALOG Function displays selected information from the current CATALOG File, the UPDATE Function updates the contents of the current CATALOG File for a particular disk, and the STATUS/ENVIRONMENT Function displays the current default settings of the CAT program and allows the user to change them.

The CATALOG Function

The CATALOG Function displays selected information from the current CATALOG File. This information includes the names of the selected files and the names of the selected disks they reside on. For the purpose of this discussion, let's assume that the default function for CAT is CATALOG. The CAT command by itself, then, would display the contents of the entire CATALOG File. This listing would include the names of all files in alphabetical order followed by the names of all the disks they reside on in alphabetical order. See the last chapter for examples.

With the default function set to CATALOG, the general form for CAT would be

CAT filename.typ dskname.typ /o

where 'filename.typ' and 'dskname.typ' may contain wild cards and '/o' is zero or more options. Examples of valid forms and their meanings are --

-- CAT *.COM -- display all .COM files and the disks they reside on

-- CAT *.COM *.A?? -- display all .COM files on disks whose type begins with the letter A

-- CAT *.COM *.A?? /N -- display all files which are NOT .COM files on disks whose type begins with the letter A

— CAT S*.MAC MACRO80.* -- display all files beginning with the letter S and having a type of .MAC on all disks whose name is MACRO80

As the user can note, the third example employed the '/N' option; this option negates the file name selection. That is, if a match is found and /N is on, then it is declared to be a no match; otherwise, a match occurs.

Command options which affect the CATALOG Function are —

— /C — force the CATALOG Function; the CATALOG Function is engaged regardless of what the default function is

— /F <delim> <filename> <delim> — declare the name of the CATALOG File to be 'filename.CAT'; this forces the CATALOG function to be performed on the specified CATALOG File

— /L — toggle listing of selected files; if listing is on, the output from the CATALOG Function will appear on the user's LST: device. This toggles the default listing setting; that is, if listing is engaged by default, this disengages it, and vice-versa.

— /N — negate selection. Select those files which do NOT match the 'filename.typ' wild card.

— /T — toggle default CATALOG/UPDATE Function; if UPDATE is on by default, CATALOG will be engaged, and vice-versa.

Examples of valid CAT commands are —

```
CAT *.COM /N/F'CPMUG'/C
  — CATALOG Function, Negate Selection, and CATALOG
     files on CPMUG.CAT

CAT *.ASM *.A?? /L/F/MASTER/
  -- CATALOG Function (assumed), toggle listing, and
     CATALOG files on MASTER.CAT

CAT -*.*
  -- CATALOG Function (assumed); since all files beginning
     with a '-' are selected, then only the names of
     the disks will be displayed
```

The UPDATE Function

The UPDATE Function updates the contents of the current CATALOG File for a particular disk. Under the normal mode of operation, the CATALOG File and the CAT program reside on the disk on drive A: and the disk to be

updated resides on drive B:, although this can be changed. The general form of the CAT command as used for the UPDATE Function is --

CAT x:filename.typ /o

where 'x:' may be used to specifically identify the drive on which the disk to be updated resides, 'filename.typ' specifies which files are to be entered into the CATALOG File, and '/o' are options pertaining to the UPDATE Function. Valid forms of this command are --

— CAT A: — UPDATE all files on drive A:

— CAT *.COM /N — UPDATE all files which are NOT .COM files on drive B:

— CAT /L — UPDATE all files on drive B: and list the files as they are being updated

The following are the options valid under the UPDATE Function and their meanings --

— /F <delim> <filename> <delim> -- UPDATE on the specified CATALOG File (like the /F option for CATALOG)

-- /L — toggle listing of selected files. If the listing is on, then the names of the files selected for the UPDATE will be displayed with a count of the number of files, and, as the UPDATE progresses, the names of files not previously in the CATALOG File are displayed in the form

Add: filename.typ

and the names of files deleted from the CATALOG File because they are no longer on this disk are displayed in the form

Delete: filename.typ

The /L flag toggles the default listing mode; that is, if listing is ON by default, /L turns it OFF, and vice-versa.

— /N — negate selection. This specifies that a match is NOT to be made if the wild card filename.typ specification matches as in the CATALOG Function.

— /T — toggle default CATALOG/UPDATE Function. If the default function is CATALOG, then the function to be performed will be UPDATE, and vice-versa.

— /U — force the UPDATE Function to be performed. If there is a conflict and both /U and /C are specified, then the option specified last will take affect.

The STATUS Function

The STATUS/ENVIRONMENT Function displays the current default settings of the CAT program and allows the user to change them. The default settings affected by this function are --

— The name of the CATALOG File

-- The default CAT Function (CATALOG or UPDATE)

-- The default setting for the listing (/L)

The CAT options under the STATUS/ENVIRONMENT Function are --

— /C — set the function to CATALOG

— /E — display the ENVIRONMENT of the CAT program on the user's console and return to CP/M without performing a change to the CAT program on disk. If any options such as /C or /U are specified in the same command line, the resulting environment is displayed.

— /F <delim> <filename> <delim> -- set the name of the CATALOG File

— /L — toggle the listing flag

— /S — save the current ENVIRONMENT of CAT on disk and return to CP/M. After the save is complete, the new environment is displayed to the user as in the /E option.

— /T — toggle the current default CATALOG/UPDATE Function

-- /U — set the function to UPDATE

The /E and /S commands specifically identify the STATUS Function. When any CAT command line contains either of these options, then the STATUS Function is engaged and the normal CATALOG or UPDATE is not performed. If both options are specified, the /S takes precedence.

Examples of the use of the CAT program with the STATUS Function are --

— CAT /F'MASTER'/S — change the name of the default CATALOG File to 'MASTER.CAT' and save the change on disk

— CAT /E — display the current environment of CAT

— CAT /C/S/F/CPMUG/ -- change the name of the default CATALOG File to 'CPMUG.CAT', set the default function to CATALOG, and save the change on disk

— CAT /L/U/E — toggle the listing flag, set the default function to UPDATE, and display the new environment without changing it on

disk. This is useful in seeing what an environment would be before changing it on disk.

CHAPTER 4

CAT Error Messages

Informative Messages

++ CATALOG Function ++
++ CATALOG Function Complete ++
These messages indicate when the CATALOG Function is beginning and is completed.

++ UPDATE Function ++
++ UPDATE Function Complete ++
These messages indicate when the UPDATE Function is beginning and is completed.

Delete: <filename.typ>
Add: <filename.typ>
These messages appear during an UPDATE with Listing ON, and they indicate that the specified file has been Deleted from or Added to the CATALOG File.

Selected Files --
nnn Files Selected
These messages appear during an UPDATE with Listing ON, and the first message preceeds the names of the files selected for updating and the second message follows this list and gives a count of the number of files in the list. This list includes the names of the files excluded by the file exclusion part of the CATALOG File if they were selected.

++ STATUS Function ++
This message indicates when the STATUS Function is beginning.

Listing of Selected Files is ON
Listing of Selected Files is OFF
These messages appear in a STATUS Function display and indicate the status of the Listing Flag.

Default Function: UPDATE
Default Function: CATALOG
These messages appear in a STATUS Function display and indicate the default function which is currently set.

CATALOG File: <filename.CAT>
This message appears under several conditions and indicates the name of the CATALOG File.

Change Complete
This message appears when an /S option is invoked and the new environment of CAT is successfully copied back to disk.

No Matching Entries in Directory
This message indicates that no entries were selected.

++ CAT Interrupted ++
This message appears when the user types an <ESC> or CTRL-C during a CAT Function. If this occurs during an UPDATE, then the current CATALOG File is in its backup state and has the name 'CAT.$$$' and should be renamed; if this occurs during a CATALOG, no problems should be encountered.

Type CTRL-C to Abort or anything else to Continue -
Self-explanatory.

Please set Top of Form --
Self-explanatory. This messages occurs when Listing is engaged for the CATALOG Function.

Fatal Error Messages

Invalid Drive Specification
Self-explanatory.

CATALOG Name too Long -- Aborting
The name given for the catalog file has more than 8 characters in the filename portion or 3 characters in the filetype portion (filename.typ).

FILE ERROR -- Returning to CP/M
An error occurred while trying to open, close, read, or write to the CATALOG File. Check to see that it is intact and has its proper name (is not named 'CAT.$$$').

ERROR in CATALOG File Name Description -- Aborting
See 'CATALOG Name too Long -- Aborting' above.

ERROR -- File Output Error to CATALOG
A write error occurred while performing an UPDATE. Possible causes include running out of disk or directory space.

ERROR -- Invalid CATALOG Structure
ERROR ON: <filename.CAT>
The structure of the CATALOG File is not valid. Possible causes include not specifying the omitted files or just an empty () if there are none.

ERROR -- Error in closing CATALOG File
Self-explanatory. The backup 'CAT.$$$' should still exist.

ERROR -- CATALOG Format Error -- Premature EOF
See 'ERROR -- Invalid CATALOG Structure' above.

ERROR -- CATALOG Format Error -- No Beginning '('
See 'ERROR -- Invalid CATALOG Structure' above.

ERROR -- No Directory Space for New CATALOG
Self-explanatory.

ERROR -- No Name Present on Disk
An UPDATE Function was attempted on a disk which did not have a valid

disk name as the first entry of its sorted directory. Possible causes include the name not being present and another file containing a character less than '-' in the ASCII collating sequence as the first character of its name.

ERROR — Cannot Open CATALOG File

Self-explanatory. Check to see that the required CATALOG File resides on disk.

CHAPTER 5

Sample Terminal Session

The following is a series of screen displays illustrating the usage of the CAT program. These displays are almost exact copies of the output from CAT as it would appear on the user's console; the differences are very minor.

The lines starting with 'A>' indicate information typed by the user (as per a normal CP/M terminal session), and the rest of the information is that produced by the result of the commands typed.

```
A>XDIR B:

XDIR V1.6


FILENAME.TYP SIZE   FILENAME.TYP SIZE   FILENAME.TYP SIZE
------------ ---- ------ -------- --- ------ -------- --- ----
-SYSTEM .-06    0K   FORMAT   .COM   2K   SYSGEN   .COM   2K
USER32  .ASM   18K   HELP     .COM   2K   TERM     .COM   6K
USER32B .ASM   18K   LIST     .COM   2K   WM       .COM  10K
USER48  .ASM   18K   LOAD     .COM   2K   WS       .COM  26K
USER56  .ASM   18K   MEMTEST  .COM   2K   WSMSGS   .COM  20K
ABORTSUB.COM    2K   MOVCPM   .COM  10K   XDIR     .COM   6K
ASM     .COM    8K   MSBFIX   .COM   2K   ZDT      .COM   6K
COPY    .COM    2K   NAMEDISK.COM   2K   ZRUN     .COM   2K
CPM56   .COM   10K   PIP      .COM   8K   CPM      .HLP  32K
DDT     .COM    6K   PRINT    .COM   4K   HELP     .HLP   8K
DENSITY .COM    2K   RESOURCE.COM   6K   INDEX    .HLP  10K
DUTIL   .COM    5K   S        .COM   2K   WM       .HLP   4K
ED      .COM    6K   SINGLE   .COM   2K   NEWDISK  .SUB   2K
EDIT80  .COM   14K   STAT     .COM   4K   DISKS    .TXT   2K
ERASE   .COM    4K   SUBMIT   .COM   2K
 51 ENTRIES & 44 FILES IN DIRECTORY — 162K BYTES REMAINING
A>CAT /?    ; display HELP information

CATALOG V1.1
CATALOG COMMAND —
  CAT x:filename.typ dskname.dsk /o
 All arguments are optional, and wild cards (*,?) are
 permitted in "filename.typ" and "diskname.dsk".
Options are —
        /C = invoke CATALOG Function on A:
        /E = display CATALOG Environment on CON:
        /F <delim> <FILENAME> <delim> = declare
             name of CATALOG File 'FILENAME.CAT';
             at most 8 characters may be in FILENAME
        /L = toggle listing of selected files
        /N = negate selection
             If this option is given, wild card
             specifies files NOT to be selected
        /S = save CATALOG Environment on disk
        /T = toggle default CATALOG/UPDATE Function
        /U = invoke UPDATE Function on B: (or x:)
```

```
A>CAT /E

CATALOG V1.1
++ STATUS Function ++
  CATALOG File:     MASTER  .CAT
  Default Function: CATALOG
Listing of Selected Files is OFF
A>CAT /T/S

CATALOG V1.1
Change Complete
++ STATUS Function ++
  CATALOG File:     MASTER  .CAT
  Default Function: UPDATE
Listing of Selected Files is OFF
A>CAT /E

CATALOG V1.1
++ STATUS Function ++
  CATALOG File:     MASTER  .CAT
  Default Function: UPDATE
Listing of Selected Files is OFF
A>CAT /T/S/L

CATALOG V1.1
Change Complete
++ STATUS Function ++
  CATALOG File:     MASTER  .CAT
  Default Function: CATALOG
Listing of Selected Files is ON
A>CAT /L/S

CATALOG V1.1
Change Complete
++ STATUS Function ++
  CATALOG File:     MASTER  .CAT
  Default Function: CATALOG
Listing of Selected Files is OFF
```

```
A>CAT CAT.ASM

CATALOG V1.1
Catalog File:          MASTER  .CAT
++ CATALOG Function ++
Filename.Typ   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk
CAT     .ASM   ASM     .A00   BACK4   .A09   SRC4    .A08
++ CAT Interrupted ++
A>CAT CAT.*

CATALOG V1.1
Catalog File:          MASTER  .CAT
++ CATALOG Function ++
Filename.Typ   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk
CAT     .ASM   ASM     .A00   BACK4   .A09   SRC4    .A08
CAT     .COM   HELP    .-08   HELP    .D06   MASTER  .A04
++ CAT Interrupted ++
A>CAT CAT.* *.A??

CATALOG V1.1
Catalog File:          MASTER  .CAT
++ CATALOG Function ++
Filename.Typ   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk
CAT     .ASM   ASM     .A00   BACK4   .A09   SRC4    .A08
CAT     .COM   MASTER  .A04
++ CAT Interrupted ++
A>CAT *.* ASM.A??

CATALOG V1.1
Catalog File:          MASTER  .CAT
++ CATALOG Function ++
Filename.Typ   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk
-ASM    .A00   ASM     .A00
ASM     .COM   ASM     .A00
ASM     .HLP   ASM     .A00
ASM     .SUB   ASM     .A00
CAT     .ASM   ASM     .A00
CHESSPAT.ASM   ASM     .A00
CIO     .ASM   ASM     .A00
CKSUM   .ASM   ASM     .A00
COMPARE .ASM   ASM     .A00
DUTIL   .ASM   ASM     .A00
ED      .COM   ASM     .A00
FILEO   .ASM   ASM     .A00
GAMEPAT1.ASM   ASM     .A00
GAMEPAT2.ASM   ASM     .A00
HELP    .ASM   ASM     .A00
HELP    .COM   ASM     .A00
++ CAT Interrupted ++
A>CAT -*.*

CATALOG V1.1
Catalog File:          MASTER  .CAT
++ CATALOG Function ++
```

```
Filename.Typ   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk
-ALGOL  .B05   ALGOL   .B05
-ASM    .A00   ASM     .A00
-BACK1  .A03   BACK1   .A03
-BACK2  .B00   BACK2   .B00
-BACK3  .C05   BACK3   .C05
-BACK4  .A09   BACK4   .A09
-BACK5  .D09   BACK5   .D09
-BASICE .B09   BASICE  .B09
-C      .A02   C       .A02
-CBASIC .B08   CBASIC  .B08
-COBOL  .C05   COBOL   .C05
-COMP   .A01   COMP    .A01
-COMP   .C08   COMP    .C08
-COMP   .D02   COMP    .D02
-COMPBAK.D03   COMPBAK .D03
-COMPBAK.D04   COMPBAK .D04
-COMPBAK.D05   COMPBAK .D05
-CPM32  .A06   CPM32   .A06
-CPM32  .A07   CPM32   .A07
-FORTH  .D00   FORTH   .D00
-FORTRAN.B07   FORTRAN .B07
Type CTRL-C to Abort or anything else to Continue -
Filename.Typ   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk
-GAMES  .-07   GAMES   .-07
-HELP   .-08   HELP    .-08
-HELP   .D06   HELP    .D06
-MAC    .D01   MAC     .D01
-MACRO80.C00   MACRO80 .C00
-MACRO80.C02   MACRO80 .C02
-MASTER .A04   MASTER  .A04
-MBASIC .C07   MBASIC  .C07
-PASMT  .A05   PASMT   .A05
-PASZ   .C09   PASZ    .C09
-PERSONL.-04   PERSONL .-04
-SMAL   .C01   SMAL    .C01
-SRC1   .B04   SRC1    .B04
-SRC2   .B05   SRC2    .B05
-SRC3   .C04   SRC3    .C04
-SRC4   .A08   SRC4    .A08
-SRC5   .D03   SRC5    .D08
-STOIC  .D07   STOIC   .D07
-SYSTEM .-06   SYSTEM  .-06
-SYSTEM .B03   SYSTEM  .B03
-TEXT1  .B01   TEXT1   .B01
Type CTRL-C to Abort or anything else to Continue -
Filename.Typ   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk
-TEXT2  .B02   TEXT2   .B02
-TEXT3  .C03   TEXT3   .C03
++ CAT Interrupted ++
A>CAT S*.MAC MACRO.*

CATALOG V1.1
Catalog File:          MASTER  .CAT
```

```
++ CATALOG Function ++
Filename.Typ   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk

++ CATALOG Function Complete ++
A>CAT S*.MAC MACRO*.*

CATALOG V1.1
Catalog File:          MASTER  .CAT
++ CATALOG Function ++
Filename.Typ   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk
S       .MAC   MACRO80 .C00
S1FILEIO.MAC   MACRO80 .C02
S2FILEIO.MAC   MACRO80 .C02
S3FILEIO.MAC   MACRO80 .C02
SBDOS   .MAC   MACRO80 .C02
SCAPS   .MAC   MACRO80 .C02
SCCOUT  .MAC   MACRO80 .C02
SCIN    .MAC   MACRO80 .C02
SCIO    .MAC   MACRO80 .C02
SCLOUT  .MAC   MACRO80 .C02
++ CAT Interrupted ++
```

```
A>CAT /U

CATALOG V1.1
Catalog File:          MASTER  .CAT
++ UPDATE Function ++
 18 Files Selected
++ UPDATE Function Complete ++
A>CAT /U/L

CATALOG V1.1
Catalog File:          MASTER  .CAT
++ UPDATE Function ++
Selected Files --
     -GAMES  .-07     CHESS   .COM     CHESS   .ZER     HURKLE  .BAS
     LIFE    .ZER     LIFE8   .COM     MBASIC  .COM     PIP     .COM
     PONG    .COM     PTRN    .ZER     REVERSE .BAS     ROCKET  .BAS
     TARG    .ZER     TIMESQ  .COM     TREK80  .ZER     XDIR    .COM
     ZING    .ZER     ZRUN    .COM

 18 Files Selected
++ UPDATE Function Complete ++
A>CAT /T/S

CATALOG V1.1
Change Complete
++ STATUS Function ++
  CATALOG File:     MASTER  .CAT
  Default Function: UPDATE
Listing of Selected Files is OFF
A>CAT *.COM /N/L

CATALOG V1.1
Catalog File:          MASTER  .CAT
++ UPDATE Function ++
Selected Files --
     -GAMES  .-07     CHESS   .ZER     HURKLE  .BAS     LIFE    .ZER
     PTRN    .ZER     REVERSE .BAS     ROCKET  .BAS     TARG    .ZER
     TREK80  .ZER     ZING    .ZER

 10 Files Selected
Delete:     CHESS   .COM
Delete:     LIFE8   .COM
Delete:     MBASIC  .COM
Delete:     PONG    .COM
Delete:     TIMESQ  .COM
Delete:     ZRUN    .COM
++ UPDATE Function Complete ++
A>CAT *.ZER /N/L

CATALOG V1.1
Catalog File:          MASTER  .CAT
++ UPDATE Function ++
Selected Files --
     -GAMES  .-07     CHESS   .COM     HURKLE  .BAS     LIFE8   .COM
```

```
     MBASIC  .COM     PIP     .COM     PONG    .COM     REVERSE .BAS
     ROCKET  .BAS     TIMESQ  .COM     XDIR    .COM     ZRUN    .COM

 12 Files Selected
Add:         CHESS   .COM
Delete:      CHESS   .ZER
Delete:      LIFE    .ZER
Add:         LIFE8   .COM
Add:         MBASIC  .COM
Add:         PONG    .COM
Delete:      PTRN    .ZER
Delete:      TARG    .ZER
Add:         TIMESQ  .COM
Delete:      TREK80  .ZER
Delete:      ZING    .ZER
Add:         ZRUN    .COM
++ UPDATE Function Complete ++
A>CAT /L

CATALOG V1.1
Catalog File:          MASTER  .CAT
++ UPDATE Function ++
Selected Files --
     -GAMES  .-07     CHESS   .COM     CHESS   .ZER     HURKLE  .BAS
     LIFE    .ZER     LIFE8   .COM     MBASIC  .COM     PIP     .COM
     PONG    .COM     PTRN    .ZER     REVERSE .BAS     ROCKET  .BAS
     TARG    .ZER     TIMESQ  .COM     TREK80  .ZER     XDIR    .COM
     ZING    .ZER     ZRUN    .COM

 18 Files Selected
Add:         CHESS   .ZER
Add:         LIFE    .ZER
Add:         PTRN    .ZER
Add:         TARG    .ZER
Add:         TREK80  .ZER
Add:         ZING    .ZER
++ UPDATE Function Complete ++
A>CAT /T/S

CATALOG V1.1
Change Complete
++ STATUS Function ++
  CATALOG File:     MASTER  .CAT
  Default Function: CATALOG
Listing of Selected Files is OFF
A>CAT *.* GAMES.*

CATALOG V1.1
Catalog File:          MASTER  .CAT
++ CATALOG Function ++
Filename.Typ   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk
-GAMES  .-07   GAMES   .-07
CHESS   .COM   GAMES   .-07
CHESS   .ZER   GAMES   .-07
```

```
HURKLE   .BAS   GAMES   .-07
LIFE     .ZER   GAMES   .-07
LIFE3    .COM   GAMES   .-07
MBASIC   .COM   GAMES   .-07
PONG     .COM   GAMES   .-07
PTRN     .ZER   GAMES   .-07
REVERSE  .BAS   GAMES   .-07
ROCKET   .BAS   GAMES   .-07
TARG     .ZER   GAMES   .-07
TIMESQ   .COM   GAMES   .-07
TREK80   .ZER   GAMES   .-07
ZING     .ZER   GAMES   .-07
ZRUN     .COM   GAMES   .-07
++ CATALOG Function Complete ++
```

```
*  PROGRAM NAME:  CATALOG
*  AUTHOR:  RICHARD CONN
*  DATE:  25 JUNE 1980
*  VERSION:  1.1
*  PREVIOUS VERSION:  1.0 (15 JUNE 1980)

*
*  CATALOG PROGRAM -- CAT.ASM
*
*       The purpose of this program is to provide the user with a convenient
* way to keep track of all or a selected subset of his files on all or a
* selected subset of his disks.
*
*       CAT does this by performing two basic functions -- CATALOG and UPDATE.
* The CATALOG Function allows the user to review the contents of a specified
* or implied catalog file on either CON: or LST:.  He may use wild card
* specifications to select only certain files on certain disks.  The UPDATE
* Function allows the user to update the contents of a specified or implied
* catalog file.  Again, he may use wild card specifications to select only
* certain files to be included in the update.
*
*       ** General Form of Command **
*               CAT X:FILENAME.TYP DISKNAME.TYP /O
*         Only CAT is required; all others are optional.  Wild Cards (?,*)
* may be included as part of the file name or disk name specifications.
*
*       ** Switches **
*       CAT responds to the following switches --
*               /C -- Perform the CATALOG Function (CATALOG assumed to be on
*                     drive A:)
*               /U -- Perform the UPDATE Function (Disk to be updated assumed
*                     to be on drive B:)
*               /T -- Toggle the current default function and perform
*                       it (i.e., if the default function is UPDATE,
*                       CATALOG will be performed, and vice-versa)
*
*               /F <delim> <FILENAME> <delim> -- Change the name of the
*                       Catalog File to <FILENAME>; <delim> may be any
*                       valid delimiter, like / or '; only the FILENAME
*                       is specified, and '.CAT' is the assumed file type;
*                       examples:  /F'MASTER2', /F/MAST/
*
*               /N -- Negate selection; for UPDATE, select all files which
```

```
*                   do NOT match the specified or implied wild card (if
*                   no wild card or file specification is given, no
*                   files are selected, thereby effectively deleting
*                   all files for a specified disk from the catalog);
*                   for CATALOG, select all files which do NOT match the
*                   specified or implied wild cards (one wild card for
*                   the file names and the other for the disk names);
*                   this function may be made a default, in which case
*                   normal selection is engaged (see /E and /S below)
*
*           /L -- List/Print Selected Entries; for UPDATE, list on the
*                   CON: the names of the selected files from the
*                   disk whose entries are being updated and specify
*                   all deletions and additions being made for this
*                   disk; for CATALOG, print selected entries on LST:
*                   (output normally goes to CON:); this function may
*                   be made a default, in which case /L does the reverse
*
*           /E -- Display the current environment of CAT; this command
*                   displays to the user the name of the currently-defined
*                   Catalog File, the currently-defined function (CATALOG
*                   or UPDATE), and whether /L and /N are currently
*                   engaged; the current function is not performed and
*                   control is returned to CP/M
*
*           /S -- Save the current environment as the new default
*                   environment of CAT; upon completion, control is
*                   returned to CP/M; by using the options above in
*                   conjunction with this command, the user may
*                   change the default environment of CAT; examples:
*                   CAT /C /L /F'MAST' /S -- changes the default
*                     function to CATALOG, sets listing on LST: as
*                     default List Function (assuming listing is
*                     currently off; otherwise, listing is on CON:),
*                     and changes the default Catalog File name to
*                     MAST.CAT; now, typing just CAT will result in
*                     a catalog listing of MAST.CAT to be printed on
*                     the LST: device
*                   CAT /T /S -- toggles the current function; if it
*                     is UPDATE, function is now CATALOG, and vice-versa
*                          /S always performs a /E before returning to
*                   CP/M, and /E should NOT be used in conjunction with
*                   /S
```

```
*
*  Examples --
*       CAT *.COM
*               If default function is UPDATE, all *.COM files on drive B:
* are selected and placed in the default Catalog File on drive A: (assuming
* the user is logged in to drive A:)
*               If default function is CATALOG, all *.COM files for all
* disks are displayed (assuming /N is not in effect)
*       CAT *.COM /N
*               Same as above, but all files other than *.COM are selected
*       CAT *.COM *.A??
*               For UPDATE, same as above.  For CATALOG, same as above but
* disks *.A?? (A-Series) are selected.
*
*  NOTE:  Disks to be cataloged must be NAMED.  A disk name is simply
*       an entry for a file which need not occupy any disk space which
*       begins with a '-', like '-DD1.-01' or '-GAMES.FUN' are valid
*       disk names.  The NAMEDISK program may be run to name a disk.
*
*  NOTE:  Due to the nature of CAT, file names starting with a character
*       (such as $) whose ASCII value is less than that of - are NOT
*       permitted.  A CAT error message will result if such files are
*       selected.
*



*
*  START OF PROGRAM
*


*
*  USER CUSTOMIZED I/O PARAMETERS
*
CRT$LINES       EQU     24      ; NUMBER OF LINES ON CON:
LST$LINES       EQU     51      ; NUMBER OF LINES ON LST:

        ORG     100H
        JMP     START   ; GO TO START OF PROGRAM
*  THE FOLLOWING IS THE DEFAULT OPTION FOR UPDATE OR CATALOG FUNCTION
UPDATE$FLAG:
        DB      0       ; SET FOR CATALOG FUNCTION
*  THE FOLLOWING IS THE FILE LIST FLAG
LIST$FLAG:
```

```
        DB      0       ; SET FOR NO LISTING
*   HE FOLLOWING IS THE DEFAULT NAME FOR THE CATALOG FILE
FCB1:   DB      0,'MASTER  CAT',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
*  THE FOLLOWING IS THE NAME OF THIS PROGRAM
FCB2:   DB      0,'CAT     COM',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
*  THE FOLLOWING IS THE NAME OF THE TEMPORARY CATALOG DATA FILE
FCB3:   DB      0,'$CAT$   $$$',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
*  THE FOLLOWING IS THE DUAL FCB USED BY BOTH CATALOG AND UPDATE FUNCTIONS
DUAL$FCB:
        DB      0,'            ',0,0,0,0
DUAL$FCB1:
        DB      0,'            ',0,0,0,0


START:
        LXI     SP,STACK        ; RESET STACK
        CALL    PRINT$MESSAGE
        DB      'CATALOG V1.1',0

*
*  MOVE DUAL FCB INTO HOLDING AREA
*
        LXI     H,FCB   ; PT TO FCB
        LXI     D,DUAL$FCB      ; PT TO DUAL BUFFER
        MVI     B,32    ; 32 BYTES
        CALL    MOVE

*
*  MOVE FCB AND BUFF TO TFCB AND TBUFF
*
        LXI     H,FCB   ; MOVE FCB
        LXI     D,TFCB
        MVI     B,16    ; 16 BYTES
        CALL    MOVE
        LXI     H,BUFF  ; MOVE BUFF
        LXI     D,TBUFF
        MVI     B,128   ; 128 BYTES
        CALL    MOVE
*
*  PLACE 0 AT END OF COMMAND STRING
*
        LXI     H,TBUFF ; PT TO COMMAND STRING
        MOV     A,M     ; GET CHAR CNT
```

```
        ADD     L       ; PT TO LAST BYTE
        MOV     L,A
        INR     L       ; PT TO AFTER LAST BYTE
        MVI     M,0     ; STORE ZERO
*
*  EXTRACT DRIVE SPECIFICATION, IF ANY
*
        LXI     H,TBUFF+1       ; SCAN FOR DRIVE SPEC
DSPEC   MOV     A,M     ; SKIP TO NON-BLANK
        INX     H       ; PT TO NEXT
        CPI     ' '
        JZ      DSPEC
        ORA     A       ; EOLN?
        JZ      DRIVEB  ; SELECT B:
        MOV     A,M     ; PTING TO 2ND CHAR — GET IT
        CPI     ':'     ; DRIVE SPEC?
        JNZ     DRIVEB  ; SELECT B:
        DCX     H       ; GET DRIVE LETTER
        MOV     A,M
        SUI     'A'     ; CONVERT TO 0-3
        CPI     4       ; ERROR?
        JC      DID
*  INVALID DRIVE SPECIFICATION
        CALL    PRINT$MESSAGE
        DB      CR,LF,'Invalid Drive Specification',0
        JMP     CPM     ; PRINT ERROR MSG AND EXIT
*  SELECT DRIVE B:
DRIVEB:
        MVI     A,1     ; 1=DRIVE B:
*  SAVE DRIVE NUMBER TO LOG IN FOR LATER
DID:
        STA     LOG$DRIVE       ; SAVE DRIVE NUMBER TO LOG IN

*
*  SCAN FOR AND PROCESS OPTIONS
*
OPTION:
        LXI     H,TBUFF+1       ; LOOK FOR OPTIONS
        XRA     A
        STA     NEGFLG  ; TURN OFF NEGATION FLAG
        STA     CHNGFLG ; TURN OFF CHANGE OPTION
OPT:
        MOV     A,M     ; OPTION?
```

```
        INX     H       ; PT TO NEXT
        CPI     OPTC
        JZ      OPT1
        ORA     A       ; EOLN?
        JNZ     OPT
        LDA     CHNGFLG ; CHANGE OPTION INVOKED?
        ORA     A       ; 0=NO
        JNZ     CHANGE  ; CHANGE IF YES
        JMP     PRINT$CATALOG$NAME      ; PRINT NAME OF CATALOG FILE AND CONT

*  OPTION DETECTED
OPT1:
        MOV     A,M     ; GET OPTION CHAR
        CPI     'N'     ; NEGATE SELECTION?
        JZ      OPTNEG
        CPI     'F'     ; DECLARE NEW NAME FOR CATALOG FILE
        JZ      OPTFILE
        CPI     'S'     ; SAVE ENVIRONMENT AS DEFINED ON DISK
        JZ      OPTCHNG
        CPI     'C'     ; FORCE CATALOG
        JZ      OPTFCAT
        CPI     'U'     ; FORCE UPDATE
        JZ      OPTFUPDATE
        CPI     'T'     ; TOGGLE UPDATE FLAG
        JZ      OPTUPDATE
        CPI     'E'     ; DISPLAY ENVIRONMENT OF PROGRAM
        JZ      OPTSTAT
        CPI     'L'     ; TOGGLE LISTING FLAG
        JZ      OPTLIST
*  INVALID OPTION
        LXI     D,HELPMS        ; PRINT HELP MESSAGE
        MVI     C,9     ; PRINT ON CON:
        CALL    BDOS
        JMP     CPM

*
*  FORCE CATALOG FUNCTION
*
OPTFCAT:
        MVI     A,0     ; UPDATE OFF
        STA     UPDATE$FLAG
        INX     H       ; PT TO NEXT
        JMP     OPT     ; CONTINUE OPTION PROCESSING
```

```
*
*  FORCE UPDATE FUNCTION
*
OPTFUPDATE:
        MVI     A,0FFH  ; UPDATE ON
        STA     UPDATE$FLAG
        INX     H       ; PT TO NEXT
        JMP     OPT     ; CONTINUE OPTION PROCESSING

*
*  DISPLAY STATUS OF CATALOG PROGRAM
*
OPTSTAT:
        CALL    PRINT$MESSAGE
        DB      CR,LF,'++ STATUS Function ++',0
        CALL    SET$ENTRY$COUNT ; FAKE OUT PRINT ROUTINE
        CALL    PRINT$MESSAGE
        DB      CR,LF,'  CATALOG File:',0
        LXI     H,FCB1+1        ; PRINT NAME OF CATALOG FILE
        CALL    PRINT$ENTRY
        CALL    PRINT$MESSAGE
        DB      CR,LF,'  Default Function: ',0
        LDA     UPDATE$FLAG     ; DISPLAY NAME OF DEFAULT FUNCTION
        ORA     A       ; 0=CATALOG
        JZ      OPTSTAT$CAT
        CALL    PRINT$MESSAGE
        DB      'UPDATE',0
        JMP     OPTSTAT$CAT1
OPTSTAT$CAT:
        CALL    PRINT$MESSAGE
        DB      'CATALOG',0
OPTSTAT$CAT1:
        CALL    PRINT$MESSAGE
        DB      CR,LF,'Listing of Selected Files is ',0
        LDA     LIST$FLAG       ; DISPLAY LISTING OPTION
        ORA     A       ; 0=OFF
        JNZ     OPTSTAT$LIST
        CALL    PRINT$MESSAGE
        DB      'OFF',0
        JMP     OPTSTAT$DONE
OPTSTAT$LIST:
        CALL    PRINT$MESSAGE
```

```
        DB      'ON',0
OPTSTAT$DONE:
        JMP     CPM

*
*  TOGGLE UPDATE FLAG
*
OPTUPDATE:
        LDA     UPDATE$FLAG     ; GET FLAG
        CMA
        STA     UPDATE$FLAG
        INX     H       ; PT TO NEXT OPTION CHAR
        JMP     OPT     ; CONTINUE OPTION PROCESSING

*
*  TOGGLE LISTING FLAG
*
OPTLIST:
        LDA     LIST$FLAG       ; GET FLAG
        CMA
        STA     LIST$FLAG
        INX     H       ; PT TO NEXT OPTION CHAR
        JMP     OPT     ; CONTINUE OPTION PROCESSING

*
*  DECLARE NEW NAME FOR CATALOG FILE
*
OPTFILE:
        INX     H       ; PT TO DELIMITER
        MOV     A,M     ; ... IN A
        ORA     A       ; 0=DONE--ERROR
        JNZ     OPTFILE1
        CALL    PRINT$MESSAGE
        DB      CR,LF,'ERROR in CATALOG File Name Declaration -- Aborting',0
        JMP     CPM
OPTFILE1:
        MOV     C,A     ; DELIMITER IN C
        INX     H       ; PT TO 1ST CHAR
        LXI     D,FCB1+1        ; PLACE <SP> IN FILE NAME PART
        PUSH    D       ; SAVE PTR
        XCHG            ; DE PTS TO 1ST CHAR, HL PTS TO FILE NAME PART
        MVI     B,8     ; <SP> OUT FILE NAME
        MVI     A,' '   ; <SP>
```

```
        CALL    FILL
        POP     H       ; HL PTS TO FILE NAME PART, DE PTS TO 1ST CHAR
        MVI     B,8     ; AT MOST 8 CHARS
FILE$NAME:
        LDAX    D       ; GET CHAR
        CMP     C       ; DONE?
        JZ      FILE$NAME$DONE
        ORA     A       ; DONE?
        JZ      FILE$NAME$DONE
        MOV     M,A     ; PUT CHAR
        INX     H       ; PT TO NEXT
        INX     D
        DCR     B       ; COUNT DOWN
        JNZ     FILE$NAME
        LDAX    D       ; GET NEXT CHAR -- MUST BE DELIMITER
        CMP     C       ; OK?
        JZ      FILE$NAME$DONE
        CALL    PRINT$MESSAGE
        DB      CR,LF,'CATALOG Name too Long -- Aborting',0
        JMP     CPM
FILE$NAME$DONE:
        XCHG            ; HL PTS TO NEXT CHAR OF OPTION
        JMP     OPT     ; CONTINUE OPTION PROCESSING

*
*  OPTION CHANGE -- WRITE FIRST PART OF FILE BACK TO DISK
*
OPTCHNG:
        MVI     A,0FFH  ; SET FLAG FOR NOW -- CHANGE WHEN DONE
        STA     CHNGFLG
        INX     H       ; PT TO NEXT CHAR
        JMP     OPT     ; CONTINUE OPTION PROCESSING
*
*  WRITE FIRST PART OF FILE BACK TO DISK
*
CHANGE:
        LXI     D,FCB2  ; OPEN FILE
        MVI     C,15    ; OPEN
        CALL    BDOS
        CPI     0FFH    ; ERROR?
        JNZ     CHANGE1
CHANGE0:
        CALL    PRINT$MESSAGE
```

```
        DB      CR,LF,'FILE ERROR — Returning to CP/M',0
        JMP     CPM
CHANGE1:
        LXI     H,100H  ; COPY FIRST BLOCK TO DISK
        LXI     D,80H
        MVI     B,128   ; 128 BYTES
        CALL    MOVE
        LXI     D,FCB2  ; WRITE SECTOR
        MVI     C,21    ; WRITE
        CALL    BDOS
        ORA     A       ; ERROR?
        JNZ     CHANGE0
        CALL    PRINT$MESSAGE
        DB      CR,LF,'Change Complete',0
        JMP     OPTSTAT

*
*  NEGATE SELECTION OF WILD CARD
*
OPTNEG:
        MVI     A,0FFH  ; TURN ON NEGATE FLAG
        STA     NEGFLG
        INX     H       ; PT TO NEXT BYTE
        JMP     OPT     ; CONTINUE PROCESSING OPTIONS

*
*  PRINT NAME OF CATALOG FILE
*
PRINT$CATALOG$NAME:
        CALL    PRINT$MESSAGE
        DB      CR,LF,'Catalog File:    ',0
        CALL    SET$ENTRY$COUNT ; FAKE OUT PRINT ROUTINE
        LXI     H,FCB1+1        ; PT TO FILE NAME
        CALL    PRINT$ENTRY

*  SELECT UPDATE OR CATALOG FUNCTIONS
        LDA     UPDATE$FLAG     ; 0=CATALOG, 0FFH=UPDATE
        ORA     A       ; CATALOG?
        JZ      CATALOG$FUNCTION

**********************
*  UPDATE FUNCTION *
**********************
```

```
UPDATE$FUNCTION:
        CALL    PRINT$MESSAGE
        DB      CR,LF,'++ UPDATE Function ++',0
        XRA     A       ; TURN OFF PRINT
        STA     PRINT$FLAG

*
*  SET POINTERS TO INPUT (READ) AND OUTPUT (WRITE) FILES
*
        LXI     H,FCB3  ; READ
        SHLD    READ$FILE$PTR
        LXI     H,FCB1  ; WRITE
        SHLD    WRITE$FILE$PTR

*
*  MAKE FCB WILD (EQUIV TO *.*)
*
        LXI     H,TFCB  ; NAME SPECIFIED?
        MVI     M,0     ; SET FCB TO WILD
        INX     H       ; STORE NAME AND EXT AS ?
        MVI     B,11    ; 11 CHARS
        MVI     A,'?'   ; '?' IS WILD CHAR
        CALL    FILL
        MVI     B,21    ; REST OF CHARS ARE ZERO
        MVI     A,0     ; 0 IS CHAR
        CALL    FILL

*
*  LOG IN DISK DRIVE AND SAVE CURRENT DRIVE NUMBER
*
        MVI     C,25    ; GET CURRENT DRIVE NUMBER
        CALL    BDOS
        STA     LFLG    ; SAVE IT IN LOGIN FLAG
        LDA     LOG$DRIVE       ; GET DRIVE NUMBER TO LOG IN
        MOV     E,A     ; ... IN E
        MVI     C,14    ; LOG IN NEW DRIVE
        CALL    BDOS

*
*  BUILD DIRECTORY TABLE
*
DIR:
        XRA     A
```

```
        STA     ENTRY$COUNT     ; SET COUNT OF ENTRIES TO ZERO
        MVI     C,17    ; SEARCH FOR FILE
        LXI     D,TFCB  ; PT TO WILD NAME
        CALL    BDOS
        CPI     255     ; NO MATCH?
        JZ      DIR$EMPTY
DIR$LOOP:
        CALL    PUT$ENTRY       ; PLACE ENTRY IN DIR
        MVI     C,18    ; SEARCH FOR NEXT MATCH
        LXI     D,TFCB  ; PT TO WILD NAME
        CALL    BDOS
        CPI     255     ; DONE?
        JNZ     DIR$LOOP

*  LOG IN ORIGINAL DRIVE
        LDA     LFLG    ; GET DRIVE NUMBER
        MOV     E,A     ; ... IN E
        MVI     C,14    ; LOG IN DISK
        CALL    BDOS

*  ALPHABETIZE DIRECTORY
        CALL    ALPHABETIZE

*  SET FILE COUNT
        LDA     ENTRY$COUNT     ; GET ENTRY COUNT
        STA     FILE$COUNT      ; SET COUNT OF FILES

*  SEARCH DIR1 FOR SPECIFIED FILE(S), IF ANY
        CALL    SEARCH$DIR

*  ANY MATCHES?
        LDA     FILE$COUNT
        ORA     A
        JNZ     DISK$NAME$TEST

*  NO MATCHING ENTRIES
DIR$EMPTY:
        CALL    PRINT$MESSAGE
        DB      CR,LF,'No Matching Entries in Directory',0

*  LOG IN ORIGINAL DRIVE
        LDA     LFLG    ; GET DRIVE NUMBER
        MOV     E,A     ; ... IN E
```

```
        MVI     C,14    ; LOG IN DRIVE
        CALL    BDOS

*  RETURN TO CP/M
        JMP     CPM

*  DISK NAME MUST BE PRESENT
DISK$NAME$TEST:
        LDA     DIR1    ; 1ST BYTE OF DIR1 IS DISK NAME
        CPI     '-'     ; NAME STARTS WITH '-'
        JZ      DIR$PRINT       ; CONTINUE IF OK
        CALL    PRINT$MESSAGE
        DB      CR,LF,'ERROR -- No Name Present on Disk',0
        JMP     CPM

*  PRINT NAMES OF SELECTED FILES
DIR$PRINT:
        LDA     LIST$FLAG       ; LISTING ON?
        ORA     A       ; 0=NO
        JZ      DIR$PRINT$DONE

*  PRINT NAMES OF SELECTED FILES
        CALL    PRINT$MESSAGE
        DB      CR,LF,'Selected Files --',CR,LF,0
        CALL    SET$ENTRY$COUNT
        LDA     FILE$COUNT      ; A=NUMBER OF FILES
        MOV     C,A             ; ... IN C
        LXI     H,DIR1          ; PT TO DIR1
DIR$PRINT$LOOP:
        CALL    PRINT$ENTRY     ; PRINT FILE NAME
        DCR     C               ; DONE?
        JNZ     DIR$PRINT$LOOP
        LDA     ENTRY$COUNT     ; JUST START NEW LINE?
        CPI     4       ; IF SO, COUNT IS 4
        CNZ     CRLF
DIR$PRINT$DONE:
        CALL    CRLF    ; NEW LINE
        LDA     FILE$COUNT
        CALL    PRINT$VALUE     ; PRINT COUNT OF FILES
        CALL    PRINT$MESSAGE
        DB      ' Files Selected',0

*  PLACE CTRL-Z AT END OF DIR1
```

```
        LXI     H,DIR1  ; PT TO DIR
        LDA     FILE$COUNT
        MOV     C,A     ; NUMBER OF FILES IN C
DIR$END:
        CALL    ADD$11$TO$HL
        DCR     C       ; COUNT DOWN
        JNZ     DIR$END
        MVI     M,CTRLZ ; PLACE CTRL-Z

*  RENAME CATALOG FILE FOR RECREATION
        LXI     H,FCB1  ; CREATE DUAL FCB CONTAINING DUAL NAMES FOR RENAME FCT
        LXI     D,DUAL$FCB
        MVI     B,16    ; 16 BYTES
        CALL    MOVE
        LXI     H,FCB3  ; NEW CATALOG NAME
        LXI     D,DUAL$FCB1
        MVI     B,16    ; 16 BYTES
        CALL    MOVE
        LXI     D,DUAL$FCB      ; RENAME CATALOG FILE
        MVI     C,23    ; RENAME FUNCTION
        CALL    BDOS

*  OPEN OLD CATALOG FILE FOR INPUT
OPEN$OLD$CAT:
        LXI     D,FCB3  ; OPEN OLD CATALOG UNDER ITS NEW NAME
        MVI     C,15    ; OPEN FILE
        CALL    BDOS
        CPI     255     ; ERROR?
        JNZ     READ$FIRST$SECTOR
CAT$FILE$OPEN$ERR:
        CALL    PRINT$MESSAGE
        DB      CR,LF,'ERROR -- Cannot Open CATALOG File',0
        JMP     CPM

*  READ FIRST SECTOR OF OLD CATALOG FOR GET/PUT OPERATIONS
READ$FIRST$SECTOR:
        CALL    READ$SECTOR     ; USE GENERAL READ SECTOR ROUTINE

*  MAKE NEW CATALOG FILE
OPEN$NEW$CAT:
        LXI     D,FCB1  ; OPEN NEW CATALOG
        MVI     C,22    ; MAKE FILE
        PUSH    D       ; SAVE PTR TO FCB
```

```
        CALL    BDOS
        CPI     255     ; OK?
        JNZ     OPEN$NEW$CAT1
        CALL    PRINT$MESSAGE
        DB      CR,LF,'ERROR — No Directory Space for New CATALOG',0

*  CHANGE CATALOG NAME BACK
ABORT:
        LXI     D,FCB1  ; DELETE ORIGINAL FILE
        MVI     C,19    ; DELETE FILE
        CALL    BDOS
        LXI     H,FCB3  ; PT TO BACKUP NAME
        LXI     D,DUAL$FCB      ; PLACE IN 1ST PART OF DUAL
        MVI     B,16    ; 16 BYTES
        CALL    MOVE
        LXI     H,FCB1  ; PT TO ORIGINAL NAME
        LXI     D,DUAL$FCB1     ; PLACE IN 2ND PART OF DUAL
        MVI     B,16    ; 16 BYTES
        CALL    MOVE
        LXI     D,DUAL$FCB      ; RENAME
        MVI     C,23    ; RENAME FILE
        CALL    BDOS
        JMP     CPM

*  OPEN CATALOG FOR OUTPUT
OPEN$NEW$CAT1:
        POP     D       ; GET PTR TO FCB
        MVI     C,15    ; OPEN FILE
        CALL    BDOS
        LXI     H,PUT$BUFFER    ; SET PTR TO PUT BUFFER
        SHLD    PUT$BUFFER$PTR
        MVI     A,128   ; SET PUT BYTE COUNT
        STA     PUT$BYTE$CNT

*  FORMAT DISK NAME (REMOVE LEADING '-' AND MOVE OVER 1 BYTE)
        LXI     H,DIR1+1        ; COPY 7 BYTES OVER
        LXI     D,DISK$NAME     ; PT TO DISK NAME BUFFER
        MVI     B,7     ; 7 BYTES
        CALL    MOVE
        MVI     A,' '   ; PLACE TRAILING <SP>
        STAX    D       ; INTO DISK NAME BUFFER
        INX     D       ; PT TO TYPE
        MVI     B,3     ; 3 BYTES IN TYPE
```

```
        CALL    MOVE

*  ** PROCESS EXCLUDED FILES **
*  GET AND CHECK FOR LEADING '('
        CALL    GET     ; GET BYTE
        CALL    PUT     ; PUT BYTE
        CPI     '('
        JZ      EXTRACT$FILES
EXCL$FILE$ERR:
        CALL    PRINT$MESSAGE
        DB      CR,LF,'ERROR -- CATALOG Format Error -- No Beginning ''('',0
        JMP     CPM

*  EXTRACT OMITTED FILES FROM DIR1
EXTRACT$FILES:
        LDA     WORK$BYTE       ; GET LAST BYTE OBTAINED
        CPI     ')'     ; END OF FILES?
        JZ      EXTRACT$FILES$DONE
        CPI     CTRLZ   ; ERROR?
        JNZ     EXTRACT$FILES0

*  PREMATURE EOF ENCOUNTERED
PREMATURE$EOF:
        CALL    PRINT$MESSAGE
        DB      CR,LF,'ERROR -- CATALOG Format Error -- Premature EOF',0
        JMP     CPM

*  SCAN FOR FILE TO EXTRACT
EXTRACT$FILES0:
        CALL    GET$PUT$FILE    ; GET FILENAME IN FILE1$BUF
        LXI     H,DIR1+11       ; PT TO DIR TABLE
        LXI     D,FILE1$BUF
EXTRACT$FILES1:
        MOV     A,M     ; GET BYTE
        CPI     CTRLZ   ; DONE?
        JZ      EXTRACT$FILES
        CALL    CMP$ENTRY       ; COMPARE DIR1 ENTRY AGAINST FILE1$BUF
        JNZ     EXTRACT$FILES2

*  EXTRACTION MATCH -- REMOVE ENTRY
        MVI     M,0     ; REMOVE DIR1 ENTRY
        JMP     EXTRACT$FILES   ; CONTINUE EXTRACTION
```

```
* POINT TO NEXT ENTRY IN DIR1
EXTRACT$FILES2:
        CALL    ADD$11$TO$HL
        JMP     EXTRACT$FILES1

* EXTRACTION COMPLETE — FLUSH
EXTRACT$FILES$DONE:
        CALL    GET     ; GET UNTIL <LF>
        CALL    PUT
        CPI     CTRLZ
        JZ      PREMATURE$EOF
        CPI     LF
        JNZ     EXTRACT$FILES$DONE

* START UPDATE PROCEDURE
        LXI     H,DIR1  ; PT TO FIRST ENTRY OF DIR1
        XRA     A       ; A=0
        STA     CAT$LOAD$FLAG   ; SET NO CURRENT WORK$DISK/WORK$FILE

* CHECK FOR END OF MEMORY DIRECTORY
UPDATE$CAT:
        SHLD    DIR$PTR ; PTR TO DIR ENTRY
        MOV     A,M     ; GET BYTE FROM DIR1
        CPI     CTRLZ   ; DONE?
        JZ      FLUSH$CATALOG   ; COPY REST OF CATALOG FILE OVER
        ORA     A       ; NULL?
        JNZ     UPDATE$CAT0
NEXT$DIR1:
        CALL    ADD$11$TO$HL
        JMP     UPDATE$CAT

* GET NEXT FILE FROM DISK
UPDATE$CAT0:
        CALL    GET$CAT$ENTRY   ; GET CATALOG ENTRY IN WORK$FILE AND WORK$DISK
        JZ      FLUSH$DIR       ; EOF

* COMPARE ENTIRE FILE NAME/DISK NAME ENTRY WITH TARGET DIR1 ENTRY
COMPARE$CAT:
        LXI     H,DISK$NAME     ; CHECK FOR SAME DISK
        LXI     D,WORK$DISK
        CALL    CMP$ENTRY       ; COMPARE
        JNZ     DIFFERENT$DISK
```

```
*  SAME DISK NAME -- FUNCTION MUST BE ADD/DEL/NO CHANGE
        LHLD    DIR$PTR ; PT TO DIR1 ENTRY
        LXI     D,WORK$FILE     ; PT TO TARGET FILE NAME
        XCHG            ; SWITCH FOR CMP
        CALL    CMP$ENTRY       ; CARRY SET MEANS DIR1 ENTRY<TARGET

*  CHECK FOR NO CHANGE
        JZ      SAME$FILE

*  CHECK FOR DIR1 ENTRY < TARGET; IF SO, ADD DIR1
        JC      ADD$DIR$TO$CAT

*  TARGET < DIR1, SO DELETE
        LDA     LIST$FLAG       ; LIST?
        ORA     A
        JZ      UPDATE$CAT0     ; NO IF ZERO
        CALL    PRINT$MESSAGE
        DB      CR,LF,'Delete:',0
        LXI     H,WORK$FILE
        CALL    PRINT1$ENTRY    ; PRINT FILE NAME
        JMP     UPDATE$CAT0     ; GET NEXT ENTRY FROM DISK

*  NO CHANGE IN CAT ENTRY, SO WRITE TO DISK
SAME$FILE:
        CALL    PUT$CAT$ENTRY   ; WRITE WORK$FILE AND WORK$DISK TO CAT
        LHLD    DIR$PTR ; PT TO DIR ENTRY
        JMP     NEXT$DIR1       ; PT TO NEXT DIR1 ENTRY AND CONTINUE

*  ADD DIR1 TO CATALOG
ADD$DIR$TO$CAT:
        LDA     LIST$FLAG       ; LIST?
        ORA     A
        JZ      ADD$DIR1$TO$CAT ; NO IF ZERO
        CALL    PRINT$MESSAGE
        DB      CR,LF,'Add:   ',0
        LHLD    DIR$PTR ; PRINT FILE NAME
        CALL    PRINT1$ENTRY
ADD$DIR1$TO$CAT:
        LHLD    DIR$PTR ; WRITE FILE NAME
        CALL    PUT$FILE
        MVI     A,','
        CALL    PUT
        LXI     H,DISK$NAME     ; WRITE DISK NAME
```

```
        CALL    PUT$FILE
        CALL    PUT$CRLF
        LHLD    DIR$PTR ; PT TO NEXT DIR1 ENTRY

*  POINT TO NEXT DIR1 ENTRY
ADD$DIR$NEXT:
        CALL    ADD$11$TO$HL
        MOV     A,M     ; CHECK FOR NULL ENTRY
        ORA     A       ; 0=NO ENTRY
        JZ      ADD$DIR$NEXT
        CPI     CTRLZ   ; DONE?
        JZ      FLUSH$CATALOG

*  NOW POINTING TO NEXT DIR1 ENTRY
        SHLD    DIR$PTR ; SET POINTER
        JMP     COMPARE$CAT     ; COMPARE WITH CURRENTLY-LOADED ENTRY

*  DISK NAMES ARE NOT THE SAME -- DETERMINE IF ADD DIR1 ENTRY OR TARGET
DIFFERENT$DISK:
        LHLD    DIR$PTR ; PT TO DIR1 ENTRY
        LXI     D,WORK$FILE     ; COMPARE FILE NAMES
        CALL    CMP$ENTRY
        JZ      DIFFERENT$DISK1 ; SAME -- COMPARE DISK NAMES
        JNC     ADD$DIR$TO$CAT  ; DIR1 ENTRY LESS -- ADD IT

*  COPY TARGET ENTRY TO CAT AND GET NEXT
ADD$TARGET$TO$CAT:
        CALL    PUT$CAT$ENTRY   ; WRITE WORK$FILE AND WORK$DISK TO CAT
        LHLD    DIR$PTR ; PT TO DIR1 ENTRY AND RESUME
        JMP     UPDATE$CAT

*  COMPARE DISK NAMES FOR DIFFERENT DISK NAME SUBPROGRAM
DIFFERENT$DISK1:
        LXI     H,DISK$NAME
        LXI     D,WORK$DISK
        CALL    CMP$ENTRY
        JC      ADD$TARGET$TO$CAT       ; TARGET NAME IS LESS (DE<HL)
        JMP     ADD$DIR$TO$CAT

*
*  PUT <CR> <LF> ON DISK
*
PUT$CRLF:
```

```
        MVI     A,CR
        CALL    PUT
        MVI     A,LF
        JMP     PUT

*
*  FLUSH CATALOG -- COPY REST OF OLD CATALOG INTO NEW CATALOG
*
FLUSH$CATALOG:
        LDA     CAT$LOAD$FLAG   ; ENTRY IN WORK$DISK/WORK$FILE?
        ORA     A       ; 0=NO
        JNZ     FLUSH$CAT0
FLUSH$CAT:
        CALL    GET$CAT$ENTRY   ; READ INTO WORK$FILE AND WORK$DISK FROM CAT
        JZ      FLUSH$CAT1      ; EOF
FLUSH$CAT0:
        LXI     H,DISK$NAME     ; CHECK FOR LIKE DISK -- IF SO, DELETE FILE
        LXI     D,WORK$DISK
        CALL    CMP$ENTRY       ; COMPARE
        JZ      FLUSH$CAT$DELETE

*  DISK NAMES ARE DIFFERENT -- WRITE TO DISK
        CALL    PUT$CAT$ENTRY   ; WRITE WORK$FILE AND WORK$DISK TO CAT
        JMP     FLUSH$CAT       ; CONTINUE FLUSHING

*  DISK NAMES ARE SAME -- TELL USER THAT FILE IS DELETED
FLUSH$CAT$DELETE:
        LDA     LIST$FLAG       ; LIST?
        ORA     A       ; 0=NO
        JZ      FLUSH$CAT       ; CONTINUE FLUSHING
        CALL    PRINT$MESSAGE
        DB      CR,LF,'Delete:',0
        LXI     H,WORK$FILE
        CALL    PRINT1$ENTRY
        JMP     FLUSH$CAT       ; CONTINUE FLUSHING

*  FILL OUT REST OF CAT W/CTRL-Z
FLUSH$CAT1:
        LDA     PUT$BYTE$CNT    ; DONE W/LAST PUT?
        CPI     128
        JZ      FLUSH$CAT2      ; YES ... CLOSE
        MVI     A,CTRLZ ; PUT ANOTHER CTRLZ
        CALL    PUT
```

```
        JMP     FLUSH$CAT1

*  CLOSE FILE, DELETE BACKUP, AND RETURN TO CP/M
FLUSH$CAT2:
        LXI     D,FCB1  ; CLOSE CATALOG
        MVI     C,16    ; CLOSE FILE
        CALL    BDOS
        CPI     255     ; ERROR?
        JNZ     FLUSH$CAT3
        CALL    PRINT$MESSAGE
        DB      CR,LF,'ERROR -- Error in closing CATALOG File',0
        JMP     CPM

*  DELETE BACKUP
FLUSH$CAT3:
        LXI     D,FCB3  ; PT TO FCB
        MVI     C,19    ; DELETE FILE
        CALL    BDOS

*  RETURN TO CP/M
        CALL    PRINT$MESSAGE
        DB      CR,LF,'++ UPDATE Function Complete ++',0
        JMP     CPM

*
*  FLUSH DIRECTORY -- COPY REST OF DIR1 ONTO END OF CATALOG
*
FLUSH$DIR:
        LHLD    DIR$PTR ; PT TO NEXT ENTRY

*  CHECK FOR EMPTY ENTRY; SKIP IF SO
FLUSH$DIR1:
        MOV     A,M     ; GET 1ST BYTE
        ORA     A       ; 0=NO ENTRY
        JNZ     FLUSH$DIR2
        CALL    ADD$11$TO$HL
        JMP     FLUSH$DIR1

*  CHECK FOR END OF DIR1 AND FINISH IF SO
FLUSH$DIR2:
        CPI     CTRLZ   ; END OF DIR1?
        JNZ     FLUSH$DIR3
        CALL    PUT     ; PUT CTRL-Z
```

```
        JMP     FLUSH$CAT1      ; FILL OUT REST AND FINISH

*  PLACE FILE AND DISK NAMES
FLUSH$DIR3:
        LDA     LIST$FLAG       ; LIST?
        ORA     A       ; 0=NO
        JZ      FLUSH$DIR4
        PUSH    H       ; SAVE PTR TO FILE NAME
        CALL    PRINT$MESSAGE
        DB      CR,LF,'Add:   ',0
        CALL    PRINT1$ENTRY
        POP     H       ; PT TO FILE NAME
FLUSH$DIR4:
        CALL    PUT$FILE        ; PLACE FILE NAME
        MVI     A,','   ; COMMA
        CALL    PUT
        PUSH    H       ; SAVE PTR TO NEXT ENTRY IN DIR1
        LXI     H,DISK$NAME     ; STORE DISK NAME
        CALL    PUT$FILE
        POP     H       ; GET PTR
        CALL    PUT$CRLF
        JMP     FLUSH$DIR1


**********************
*  CATALOG FUNCTION  *
**********************
CATALOG$FUNCTION:
        CALL    PRINT$MESSAGE
        DB      CR,LF,'++ CATALOG Function ++',0
        LDA     LIST$FLAG       ; SET PRINT FLAG TO LIST FLAG
        STA     PRINT$FLAG

*
*  TRANSLATE WILD CARDS IN DUAL FCB
*
        LXI     H,DUAL$FCB+1    ; PT TO 1ST
        MOV     A,M     ; CHECK 1ST CHAR
        CPI     ' '     ; NO FILE NAME MEANS WILD
        JZ      WILD$DUAL$FCB
        CPI     '/'     ; OPTION CAUGHT, SO WILD ALSO
        JZ      WILD$DUAL$FCB
        CALL    SET$FCB ; SET FCB1 ACCORDINGLY
```

```
        LXI     H,DUAL$FCB+17   ; PT TO 2ND
        CALL    SET$FCB ; SET FCB2 ACCORDINGLY
        JMP     SET$CATALOG$DISK

*  MAKE BOTH FCB'S WILD
WILD$DUAL$FCB:
        MVI     B,11    ; 11 BYTES
        MVI     A,'?'   ; WILD
        CALL    FILL
        LXI     H,DUAL$FCB+17   ; PT TO 2ND
        MVI     B,11    ; 11 BYTES
        MVI     A,'?'   ; WILD
        CALL    FILL
        JMP     SET$CATALOG$DISK

*  SCAN FCB FOR '*' AND SET REST OF IT TO '?' IF FOUND
SET$FCB:
        MVI     B,8     ; 8 BYTES FOR FILE NAME
        MOV     A,M     ; CHECK 1ST
        CPI     ' '     ; MAKE WILD IF LEADING <SP>
        JZ      SET$WILD
        CPI     '/'     ; MAKE WILD IF OPTION CAUGHT
        JZ      SET$WILD
*  SCAN FILE NAME
SET$FCB1:
        MOV     A,M     ; GET BYTE
        CPI     '*'     ; REST WILD?
        JZ      SET$FCB2
        INX     H       ; SKIP
        DCR     B
        JNZ     SET$FCB1
        JMP     SET$FCB3
*  MAKE REST OF FILE NAME WILD
SET$FCB2:
        MVI     M,'?'   ; WILD
        INX     H
        DCR     B
        JNZ     SET$FCB2
*  SCAN FILE TYPE
SET$FCB3:
        MVI     B,3     ; 3 BYTES
SET$FCB4:
        MOV     A,M     ; GET BYTE
```

```
        CPI     '*'     ; WILD?
        JZ      SET$FCB5
        INX     H       ; SKIP
        DCR     B
        JNZ     SET$FCB4
        RET
*  MAKE REST OF FILE TYPE WILD
SET$FCB5:
        MVI     M,'?'   ; WILD
        INX     H
        DCR     B
        JNZ     SET$FCB5
        RET
*  SET BOTH FCB'S TO WILD
SET$WILD:
        MVI     B,11    ; 11 BYTES
        MVI     A,'?'
        CALL    FILL
        LXI     H,DUAL$FCB+17   ; FCB 2
        MVI     B,11    ; 11 BYTES
        MVI     A,'?'
        JMP     FILL

*
*  CONTINUE WITH CATALOG FUNCTION -- SET NAME OF DISK TO GET CATALOG FROM
*
SET$CATALOG$DISK:
        LXI     H,FCB1  ; PT TO CATALOG NAME FCB
        SHLD    READ$FILE$PTR
        XCHG            ; PTR TO FCB1 IN DE
        MVI     C,15    ; OPEN FILE
        CALL    BDOS
        CPI     255     ; ERROR
        JZ      CAT$FILE$OPEN$ERR
        CALL    READ$SECTOR     ; READ 1ST SECTOR FOR FURTHER GET'S

*  SKIP EXCLUDED FILES
        CALL    GET     ; GET LEADING '('
        CPI     '('
        JNZ     EXCL$FILE$ERR
SKIP$EXCL$FILES:
        CALL    GET     ; CONTINUE TO ')'
        CPI     CTRLZ   ; EOF?
```

```
        JZ      PREMATURE$EOF
        CPI     '|'
        JNZ     SKIP$EXCL$FILES
SKIP$EXCL$FILES1:
        CALL    GET     ; GET TO <LF>
        CPI     CTRLZ   ; EOF?
        JZ      PREMATURE$EOF
        CPI     LF
        JNZ     SKIP$EXCL$FILES1

*  SET LAST FILE TO NULL
        LXI     H,LAST$FILE
        MVI     B,11    ; 11 BYTES
        MVI     A,' '   ; <SP>
        CALL    FILL

*  SET ENTRY COUNT FOR DISPLAY
        CALL    SET$ENTRY$COUNT

*  SET TOP OF FORM IF PRINT
        LDA     PRINT$FLAG
        ORA     A
        JZ      CATALOG$LOOP$HDR
        CALL    PRINT$MESSAGE
        DB      CR,LF,'Please set Top of Form -- '
        DB      CR,LF,'Type CTRL-C to Abort or anything else to Continue - ',0
        CALL    CHAR$IN ; GET RESPONSE
        CPI     'C'-40H ; CTRL-C
        JZ      CPM     ; RETURN TO CP/M
        CALL    CRLF    ; NEW LINE

*  PRINT HEADER FOR CATALOG OUTPUT
CATALOG$LOOP$HDR:
        XRA     A       ; SET PAGE NUMBER
        STA     PAGE$NUM
        CALL    PRINT$CAT$HDR

*
*  MAIN LOOP FOR CATALOG FUNCTION
*
CATALOG$LOOP:
        CALL    CTRLC$CHECK     ; CHECK FOR ABORT
        CALL    GET$CAT$ENTRY   ; GET FILE NAME/DISK NAME FROM CAT
```

```
        JZ      CATALOG$LOOP$DONE
        CALL    COMP$CAT$ENTRY  ; COMPARE WITH WILD CARD
        JNZ     CATALOG$LOOP    ; NO MATCH
        CALL    PRINT$CAT$ENTRY ; PRINT ENTRY IF MATCH
        JMP     CATALOG$LOOP
CATALOG$LOOP$DONE:
        LDA     PRINT$FLAG      ; PRINT?
        ORA     A
        JZ      CATALOG$LOOP$FINIS

*  PRINT, SO EJECT PAGE
        MVI     A,CR
        CALL    LIST$OUT        ; NEW LINE
        LDA     PAGE$COUNT      ; COUNT DOWN LINES FOR PAGE EJECT
        ADI     15      ; NEXT PAGE
        MOV     B,A     ; IN B
        MVI     A,LF    ; <LF>
EJECT$LOOP:
        CALL    LIST$OUT        ; PRINT CHAR ON LST:
        DCR     B
        JNZ     EJECT$LOOP

CATALOG$LOOP$FINIS:
        CALL    PRINT$MESSAGE
        DB      CR,LF,'++ CATALOG Function Complete ++',0
        JMP     CPM

*
*  PRINT CATALOG ENTRY ON CON: OR LST:
*
PRINT$CAT$ENTRY:

*  CHECK TO SEE IF NEW FILE
        LXI     H,WORK$FILE     ; PT TO CURRENT FILE
        LXI     D,LAST$FILE     ; PT TO PREVIOUS FILE
        CALL    CMP$ENTRY       ; COMPARE
        JZ      PRINT$CAT$DISK
        LDA     ENTRY$COUNT     ; WAS PREVIOUS LINE ALREADY TERMINATED?
        CPI     4               ; RESET IF YES
        CNZ     PCRLF           ; NEW LINE W/PAGING IF NOT
        PUSH    H               ; MAKE PREVIOUS FILE=CURRENT FILE
        MVI     B,11            ; 11 BYTES
        CALL    MOVE
```

```
        POP     H
        CALL    SET$ENTRY$COUNT ; ABORT POSSIBLE NEW LINE
        CALL    CL$ENTRY$NAME   ; PRINT JUST NAME
        CALL    SET$ENTRY$COUNT ; SET FOR NEW LINES
        JMP     PRINT$CAT1$DISK ; PRINT DISK NAME

*  PRINT DISK NAME AND <SP> OVER IF NEW LINE
PRINT$CAT$DISK:
        LDA     ENTRY$COUNT     ; NEW LINE?
        CPI     4
        JNZ     PRINT$CAT1$DISK ; JUST PRINT ENTRY
        MVI     B,12            ; 12 BYTES
        MVI     A,' '
PRINT$CAT$SP:
        CALL    CON$LST$OUT
        DCR     B
        JNZ     PRINT$CAT$SP

*  PRINT DISK NAME
PRINT$CAT1$DISK:
        LXI     H,WORK$DISK     ; PT TO DISK NAME
        CALL    CL$ENTRY        ; PRINT ON CON: OR LST:
        RET

*
*  NEW LINE WITH PAGING
*
PCRLF:
        MVI     A,CR    ; NEW LINE
        CALL    CON$LST$OUT
        MVI     A,LF
        CALL    CON$LST$OUT

        LDA     PAGE$COUNT      ; COUNT DOWN LINES
        DCR     A
        STA     PAGE$COUNT
        RNZ                     ; NO PAGE

        LDA     PRINT$FLAG      ; EJECT PAGE?
        ORA     A               ; 0=CON:
        JZ      PCRLF$PAGE

        MVI     A,LF            ; <LF> DOWN
```

```
        MVI     B,10            ; 10 LINES
PCRLF$PAGE$DOWN:
        CALL    CON$LST$OUT
        DCR     B
        JNZ     PCRLF$PAGE$DOWN

*  PRINT HEADER
PCRLF$PAGE1:
        CALL    PRINT$CAT$HDR   ; PRINT HEADER
        RET

*  OUTPUT TO CON: SO GIVE USER PROMPT
PCRLF$PAGE:
        CALL    PRINT$MESSAGE
        DB      'Type CTRL-C to Abort or anything else to Continue - ',0
        CALL    CHAR$IN ; GET RESPONSE
        CPI     'C'-40H ; CTRL-C
        JZ      CPM
        JMP     PCRLF$PAGE1

*
*  PRINT ENTRY PTED TO BY HL ON CON: OR LST:
*
CL$ENTRY:
        MVI     A,' '   ; PRINT 3 LEADING SPACES
        MVI     B,3
CL$ENTRY$SP:
        CALL    CON$LST$OUT
        DCR     B
        JNZ     CL$ENTRY$SP

*  PRINT NAME AND TYPE
CL$ENTRY$NAME:
        MVI     B,8     ; 8 BYTES
        CALL    CL$FIELD
        MVI     A,'.'   ; DECIMAL
        CALL    CON$LST$OUT
        MVI     B,3     ; 3 BYTES
        CALL    CL$FIELD

*  NEW LINE?
        LDA     ENTRY$COUNT     ; COUNT DOWN
        DCR     A
```

```
        STA     ENTRY$COUNT
        RNZ
        CALL    SET$ENTRY$COUNT
        CALL    PCRLF           ; NEW LINE
        RET

*  PRINT FIELD PTED TO BY HL FOR B BYTES ON CON: OR LST:
CL$FIELD:
        MOV     A,M     ; GET BYTE
        CALL    CON$LST$OUT
        INX     H       ; PT TO NEXT
        DCR     B
        JNZ     CL$FIELD
        RET

*
*  PRINT CATALOG HEADER AND RESET PAGE COUNT
*
PRINT$CAT$HDR:
        LDA     PRINT$FLAG      ; LISTING?
        ORA     A       ; 0=NO
        JNZ     LIST$CAT$HDR

*  PRINT ON CON:
        MVI     A,CRT$LINES-3   ; SET PAGE COUNT
        STA     PAGE$COUNT
        JMP     PRINT$STD$HDR

*  PRINT ON LST:
LIST$CAT$HDR:
        MVI     A,LST$LINES-4-10        ; SET PAGE COUNT
        STA     PAGE$COUNT
        LXI     H,PAGEMS
        CALL    PRINT$STRING
        LDA     PAGE$NUM        ; PRINT PAGE NUMBER
        INR     A       ; INCREMENT
        STA     PAGE$NUM
        CALL    PRINT$VALUE
        LXI     H,TITLE$LINE    ; PRINT INPUT LINE AS HEADER ON PAGE
        CALL    PRINT$STRING
        LXI     H,TBUFF+1
        CALL    PRINT$STRING
```

```
*  PRINT STANDARD HEADER
PRINT$STD$HDR:
        LXI     H,STD$HDR       ; PRINT HEADER
        CALL    PRINT$STRING

*  RESET NAME OF LAST FILE
        LXI     H,LAST$FILE
        MVI     B,11    ; 11 BYTES
        MVI     A,' '   ; <SP> FILL
        CALL    FILL
        RET

*
*  PRINT STRING PTED TO BY HL ON CON: OR LST:
*    STRING ENDS IN 0
*
PRINT$STRING:
        MOV     A,M     ; GET BYTE
        ORA     A       ; DONE?
        RZ
        CALL    CON$LST$OUT
        INX     H       ; PT TO NEXT
        JMP     PRINT$STRING

*
*  CHECK FOR CTRL-C AND ABORT IF SO
*
CTRLC$CHECK:
        PUSH H ! PUSH D ! PUSH B ! PUSH PSW
        MVI     C,11    ; CONSOLE CHAR AVAIL?
        CALL    BDOS
        ANI     1       ; SEE LSB
        JZ      CTRLC$CHECK$DONE
        CALL    CHAR$IN ; GET CHAR
        CPI     'C'-40H ; CTRL-C?
        JNZ     CTRLC$CHECK$DONE
        CALL    PRINT$MESSAGE
        DB      CR,LF,'++ CAT Interrupted ++',0
        JMP     CPM
CTRLC$CHECK$DONE:
        POP PSW ! POP B ! POP D ! POP H
        RET
```

```
*
*  PRINT CHAR IN A ON CON: OR LST:, DEPENDING ON LIST$FLAG
*
CON$LST$OUT:
        PUSH    PSW     ; SAVE CHAR
        LDA     PRINT$FLAG      ; GET FLAG
        ORA     A       ; 0=CON:
        JZ      CON$LST$OUTC
        POP     PSW     ; GET CHAR

*  CHAR TO LST:
LIST$OUT:
        PUSH H ! PUSH D ! PUSH B ! PUSH PSW
        MOV     E,A     ; CHAR IN E
        MVI     C,5     ; LST: OUT
        CALL    BDOS
        POP PSW ! POP B ! POP D ! POP H
        RET

*  CHAR TO CON:
CON$LST$OUTC:
        POP     PSW     ; GET CHAR
        JMP     CHAR$OUT

*
*  COMPARE CATALOG ENTRY FROM DISK WITH DUAL$FCB
*    RET W/ZERO SET IF MATCH
*
COMP$CAT$ENTRY:
*  COMPARE FILE NAME
        LXI     H,WORK$FILE     ; COMPARE FILE
        LXI     D,DUAL$FCB+1
        CALL    CMP$ENTRY       ; COMPARE
        JNZ     COMP$CAT$FAIL   ; NO MATCH — CHECK FOR NEGATIVE SELECT
*  COMPARE DISK NAME
        LXI     H,WORK$DISK     ; COMPARE DISK
        LXI     D,DUAL$FCB+17
        CALL    CMP$ENTRY       ; COMPARE
        JNZ     COMP$CAT$FAIL   ; NO MATCH — CHECK FOR NEGATIVE SELECT
*  COMPARE SUCCEEDED — CHECK FOR NEGATIVE SELECT
        LDA     NEGFLG  ; GET FLAG
        ORA     A       ; 0=NO
        RET
```

```
*  COMPARE FAILED — CHECK FOR NEGATIVE SELECT
COMP$CAT$FAIL:
        LDA     NEGFLG  ; GET FLAG
        CPI     0FFH    ; 0FFH=YES
        RET


*
*  LOAD WORK BUFFERS WITH NEXT FILE AND DISK NAMES FROM DISK
*
GET$CAT$ENTRY:
        CALL    GET$FILE        ; GET FILE NAME
        CPI     CTRLZ   ; RET W/ZERO SET IF EOF
        RZ
        CPI     ','     ; CHECK FOR VALID STRUCTURE
        JNZ     CAT$STR$ERR
        LXI     H,FILE1$BUF     ; STORE FILE NAME
        LXI     D,WORK$FILE
        MVI     B,11    ; 11 BYTES
        CALL    MOVE
        CALL    GET$FILE        ; GET DISK NAME
        CPI     CR      ; CHECK FOR VALID STRUCTURE
        JNZ     CAT$STR$ERR
        LXI     H,FILE1$BUF     ; STORE DISK NAME
        MVI     B,11    ; 11 BYTES
        CALL    MOVE
        MVI     A,1     ; SET NO ZERO
        STA     CAT$LOAD$FLAG   ; MARK ENTRY IN BUFFER
        ORA     A
        RET

*  STRUCTURAL ERROR IN CATALOG
CAT$STR$ERR:
        CALL    PRINT$MESSAGE
        DB      CR,LF,'ERROR — Invalid CATALOG Structure'
        DB      CR,LF,'ERROR ON: ',0
        LXI     H,FILE1$BUF     ; PRINT TEXT AFTER WHICH ERROR OCCURS
        CALL    PRINT1$ENTRY
        JMP     ABORT

*
*  WRITE CATALOG ENTRY FROM WORK BUFFERS TO CAT
*
```

```
PUT$CAT$ENTRY:
        XRA     A       ; A=0
        STA     CAT$LOAD$FLAG   ; MARK NO ENTRY IN BUFFER
        LXI     H,WORK$FILE     ; WRITE FILE NAME
        CALL    PUT$FILE
        MVI     A,','   ; COMMA
        CALL    PUT
        CALL    PUT$FILE        ; WRITE DISK NAME
        JMP     PUT$CRLF        ; NEW LINE

*
*  LOAD NEXT FILE NAME FROM DISK INTO FILE1$BUF; WORK$BYTE BUFFER
*    RETURNS WITH CHARACTER WHICH TERMINATED SCAN
*
GET$PUT$FILE:
        LXI     H,FILE1$BUF     ; <SP> FILL BUFFER
        PUSH    H
        MVI     B,11    ; 11 BYTES
        MVI     A,' '   ; <SP>
        CALL    FILL
        POP     H       ; GET PTR TO BEGINNING OF BUFFER
        MVI     B,8     ; UP TO 8 BYTES
        CALL    GET$PUT$FILE1
        CPI     '.'     ; TYP?
        RNZ
        MVI     B,3     ; UP TO 3 BYTES

*  EXTRACT FILE NAME UNTIL DELIMITER REACHED
GET$PUT$FILE1:
        CALL    GET     ; GET BYTE
        CALL    PUT     ; PUT BYTE
        CPI     LF      ; IGNORE <LF>
        JZ      GET$PUT$FILE1
        CPI     '.'     ; DONE W/NAME?
        JZ      GET$PUT$FILE$DONE
        CPI     ','     ; DONE
        JZ      GET$PUT$FILE$DONE
        CPI     ')'     ; DONE
        JZ      GET$PUT$FILE$DONE
        CPI     CR      ; DONE
        JZ      GET$PUT$FILE$CR
        CPI     CTRLZ   ; DONE
        JZ      GET$PUT$FILE$DONE
```

```
        MOV     M,A     ; PLACE BYTE
        INX     H
        DCR     B
        JNZ     GET$PUT$FILE1
        CALL    GET     ; FULL NUMBER OF BYTES — GET NEXT ONE
        CALL    PUT
        JMP     GET$PUT$FILE1$DONE
GET$PUT$FILE$CR:
        PUSH    PSW     ; SAVE <CR>
        CALL    GET     ; GET <LF>
        CALL    PUT
        POP     PSW     ; GET <CR>
GET$PUT$FILE$DONE:
        INX     H       ; PT TO NEXT BYTE
        DCR     B       ; COUNT DOWN TO END OF BUFFER
        JNZ     GET$PUT$FILE$DONE
GET$PUT$FILE1$DONE:
        STA     WORK$BYTE       ; SAVE WORK BYTE
        RET

*
*  PUT FILE — PLACE FILE NAME AND TYPE ON DISK; HL PTS TO NAME
*
PUT$FILE:
        MVI     B,8     ; 8 BYTES IN FILE NAME (MAX)
        CALL    PUT$FILE1
        MVI     A,'.'   ; DECIMAL
        CALL    PUT
        MVI     B,3     ; 3 BYTES IN FILE TYPE (MAX)
        CALL    PUT$FILE1
        RET
*  PUT$FILE1 — PLACE UP TO 8 BYTES ON DISK; STOP IF <SP> ENCOUNTERED
PUT$FILE1:
        MOV     A,M     ; GET BYTE
        CPI     ' '     ; <SP>?
        JZ      PUT$FILE2
        CALL    PUT     ; PUT BYTE ON DISK
        INX     H       ; PT TO NEXT
        DCR     B       ; COUNT DOWN
        JNZ     PUT$FILE1
        RET
PUT$FILE2:
        INX     H       ; <SP> ENCOUNTERED — SKIP TO END OF FIELD
```

```
        DCR     B
        JNZ     PUT$FILE2
        RET

*
*  LOAD NEXT FILE NAME FROM DISK INTO FILE1$BUF; WORK$BYTE BUFFER
*    RETURNS WITH CHARACTER WHICH TERMINATED SCAN
*
GET$FILE:
        LXI     H,FILE1$BUF     ; <SP> FILL BUFFER
        PUSH    H
        MVI     B,11    ; 11 BYTES
        MVI     A,' '   ; <SP>
        CALL    FILL
        POP     H       ; GET PTR TO BEGINNING OF BUFFER
        MVI     B,8     ; UP TO 8 BYTES
        CALL    GET$FILE1
        CPI     '.'     ; TYP?
        RNZ
        MVI     B,3     ; UP TO 3 BYTES

*  EXTRACT FILE NAME UNTIL DELIMITER REACHED
GET$FILE1:
        CALL    GET     ; GET BYTE
        CPI     LF      ; IGNORE <LF>
        JZ      GET$FILE1
        CPI     '.'     ; DONE W/NAME?
        JZ      GET$FILE$DONE
        CPI     ','     ; DONE
        JZ      GET$FILE$DONE
        CPI     ')'     ; DONE
        JZ      GET$FILE$DONE
        CPI     CR      ; DONE
        JZ      GET$FILE$CR
        CPI     CTRLZ   ; DONE
        JZ      GET$FILE$DONE
        MOV     M,A     ; PLACE BYTE
        INX     H
        DCR     B
        JNZ     GET$FILE1
        CALL    GET     ; FULL NUMBER OF BYTES — GET NEXT ONE
        JMP     GET$FILE1$DONE
GET$FILE$CR:
```

```
        PUSH    PSW     ; SAVE <CR>
        CALL    GET     ; GET <LF>
        POP     PSW     ; GET <CR>
GET$FILE$DONE:
        INX     H       ; PT TO NEXT BYTE
        DCR     B       ; COUNT DOWN TO END OF BUFFER
        JNZ     GET$FILE$DONE
GET$FILE1$DONE:
        STA     WORK$BYTE       ; SAVE WORK BYTE
        RET

*
*  READ SECTOR FROM FCB3 AND SET CONSTANTS FOR GET OPERATION
*
READ$SECTOR:
        PUSH H ! PUSH D ! PUSH B
        LHLD    READ$FILE$PTR   ; GET PTR TO FCB
        XCHG                    ; ... IN DE
        MVI     C,20    ; READ
        CALL    BDOS
        ORA     A       ; SET FLAGS — Z MEANS OK
        PUSH    PSW     ; SAVE FLAGS
        LXI     H,BUFF  ; COPY TO GET BUFFER
        LXI     D,GET$BUFFER
        MVI     B,128   ; 128 BYTES
        CALL    MOVE
        LXI     H,GET$BUFFER    ; RESET PTR TO GET BUFFER
        SHLD    GET$BUFFER$PTR
        MVI     A,128   ; SET BYTE COUNT
        STA     GET$BYTE$CNT
        POP     PSW     ; GET FLAGS
        POP B ! POP D ! POP H
        RET

*
*  WRITE SECTOR TO FCB1 AND SET CONSTANTS FOR PUT OPERATION
*
WRITE$SECTOR:
        PUSH H ! PUSH D ! PUSH B
        LXI     H,PUT$BUFFER
        LXI     D,BUFF  ; COPY TO 80H BUFFER
        MVI     B,128   ; 128 BYTES
        CALL    MOVE
```

```
        LHLD    WRITE$FILE$PTR  ; GET PTR TO FCB
        XCHG                    ; ... IN DE
        MVI     C,21    ; WRITE FILE
        CALL    BDOS
        ORA     A       ; SET FLAGS -- Z MEANS OK
        JZ      WRITE$SECTOR1
        CALL    PRINT$MESSAGE
        DB      CR,LF,'ERROR -- File Output Error to CATALOG',0
        JMP     ABORT
WRITE$SECTOR1:
        LXI     H,PUT$BUFFER    ; RESET PTR TO PUT BUFFER
        SHLD    PUT$BUFFER$PTR
        MVI     A,128   ; RESET BYTE COUNT
        STA     PUT$BYTE$CNT
        POP B ! POP D ! POP H
        RET

*
*  GET BYTE FROM FILE ROUTINE
*
GET:
        PUSH    H
        LHLD    GET$BUFFER$PTR  ; GET PTR TO BUFFER
        MOV     A,M     ; GET BYTE
        INX     H       ; PT TO NEXT
        SHLD    GET$BUFFER$PTR
        STA     WORK$BYTE       ; SAVE IT TEMPORARILY
        LDA     GET$BYTE$CNT    ; COUNT DOWN
        DCR     A
        STA     GET$BYTE$CNT
        JNZ     GET1
        CALL    READ$SECTOR     ; IF BUFFER EMPTY, READ NEXT SECTOR
        JZ      GET1    ; READ OK
        MVI     A,CTRLZ ; EOF -- SO FORCE CTRL-Z
        POP     H
        RET
GET1:
        LDA     WORK$BYTE       ; GET BYTE
        POP     H
        RET

*
*  PUT BYTE INTO OUTPUT FILE
```

```
*
PUT:
        PUSH    H
        LHLD    PUT$BUFFER$PTR
        MOV     M,A     ; PUT BYTE
        INX     H       ; PT TO NEXT
        SHLD    PUT$BUFFER$PTR
        STA     WORK$BYTE       ; SAVE IT TEMPORARILY
        LDA     PUT$BYTE$CNT    ; COUNT DOWN
        DCR     A
        STA     PUT$BYTE$CNT
        JNZ     PUT1
        CALL    WRITE$SECTOR    ; IF BUFFER FULL, WRITE TO DISK
PUT1:
        LDA     WORK$BYTE       ; GET BYTE
        POP     H
        RET



*
*  SET ENTRY COUNT — SET NUMBER OF ENTRIES PRINTED ON EACH LINE
*
SET$ENTRY$COUNT:
        MVI     A,4     ; 4 ENTRIES/LINE
        STA     ENTRY$COUNT
        RET

*
*  PRINT ONE ENTRY WITHOUT <CRLF>
*
PRINT1$ENTRY:
        CALL    SET$ENTRY$COUNT ; FAKE OUT PRINT ROUTINE

*
*  PRINT ENTRY — PRINT ENTRY PTED TO BY HL (12 BYTES/ENTRY, SKIP LAST)
*
PRINT$ENTRY:
        MVI     A,' '   ; PRINT 5 LEADING <SP>S
        MVI     B,5
PRINT$SPACE:
        CALL    CHAR$OUT
        DCR     B       ; COUNT DOWN
```

```
        JNZ     PRINT$SPACE

*  PRINT FILE NAME AND TYPE WITHOUT LEADING <SP>
PRINT$ENTRY$NAME:
        MVI     B,8     ; PRINT FILE NAME
        CALL    PRINT$FIELD
        MVI     A,'.'   ; PRINT DECIMAL
        CALL    CHAR$OUT
        MVI     B,3     ; PRINT FILE TYPE
        CALL    PRINT$FIELD
        LDA     ENTRY$COUNT     ; COUNT DOWN
        DCR     A
        STA     ENTRY$COUNT
        RNZ
        CALL    SET$ENTRY$COUNT
        JMP     CRLF    ; RESET ENTRY COUNT AND NEW LINE

*
*  PRINT FIELD PTED TO BY HL; FIELD IS B BYTES LONG
*
PRINT$FIELD:
        MOV     A,M     ; GET BYTE
        CALL    CHAR$OUT
        INX     H       ; PT TO NEXT
        DCR     B       ; COUNT DOWN
        JNZ     PRINT$FIELD
        RET

*
*  ALPHABETIZE -- ALPHABETIZES DIR1; ENTRY$COUNT CONTAINS
*       THE NUMBER OF ENTRIES IN DIR1
*
ALPHABETIZE:
        LXI     H,DIR1  ; PT TO DIR
        LXI     D,DIR1+11       ; PT TO NEXT ENTRY
        LDA     ENTRY$COUNT     ; GET COUNT
        SUI     1       ; STOP IF 1 ENTRY
        RZ
        MOV     B,A     ; COUNT IN B
ALPHA1:
        PUSH H ! PUSH D
        MOV     C,B     ; SUBCOUNT
ALPHA2:
```

```
        CALL    CMP$ENTRY       ; COMPARE ENTRIES
        CC      EXCHG           ; EXCHANGE IF BACKWARD
        CALL    ADD11   ; DE=DE+11
        DCR     C
        JNZ     ALPHA2
        POP D ! POP H
        CALL    ADD11   ; DE=DE+11
        XCHG
        CALL    ADD11   ; HL=HL+11
        XCHG
        DCR     B
        JNZ     ALPHA1
        RET

*
*  ADD11 — DE=DE+11
*
ADD11:
        XCHG
        CALL    ADD$11$TO$HL
        XCHG
        RET

*
*  COMPARE DIR ENTRY PTED TO BY HL WITH THAT PTED TO BY DE;
*       NO NET EFFECT ON HL, DE, OR BC; RET W/CARRY SET MEANS DE<HL
*
CMP$ENTRY:
        PUSH B ! PUSH D ! PUSH H

*  COMPARE BY FILE NAME, FILE TYPE, AND EXTENSION (IN THAT ORDER)
CMP$FN$FT:
        MVI     B,11    ; COMPARE FN, FT
        CALL    COMP
CMP$DONE:
        POP H ! POP D ! POP B
        RET

*
*  COMP COMPARES DE W/HL FOR B BYTES; RET W/CARRY IF DE<HL
*
COMP:
        LDAX    D       ; CHECK FOR AUTOMATIC MATCH
```

```
        CPI     '?'
        JZ      COMP1
        MOV     A,M     ; ON OTHER ALSO
        CPI     '?'
        JZ      COMP1
        LDAX    D       ; COMPARE
        CMP     M
        RC
        RNZ
COMP1:
        INX     H       ; PT TO NEXT
        INX     D
        DCR     B       ; COUNT DOWN
        JNZ     COMP
        RET
*
*  EXCHG — EXCHANGE DIR ENTRY PTED TO BY HL W/THAT PTED TO BY DE
*
EXCHG:
        PUSH B ! PUSH D ! PUSH H
        MVI     B,11    ; 11 BYTES
EXCHG1:
        LDAX    D       ; GET DE BYTE
        MOV     C,M     ; GET HL BYTE
        MOV     M,A     ; STORE DE BYTE
        MOV     A,C
        STAX    D       ; STORE HL BYTE
        INX     H       ; PT TO NEXT
        INX     D
        DCR     B       ; COUNT DOWN
        JNZ     EXCHG1
        POP H ! POP D ! POP B
        RET

*
*  SELECT FILES IN DIR1 ACCORDING TO CONTENTS OF FCB
*
SEARCH$DIR:
        LXI     D,FCB+1 ; PT TO FN
        LDAX    D       ; WILD?
        CPI     ' '     ; NO FN MEANS WILD
        JZ      SEARCH$DIR0
        CPI     '/'     ; OPTION CAUGHT, SO WILD ALSO
```

```
        JNZ     SEARCH$DIR1
SEARCH$DIR0:
        MOV     H,D     ; MAKE HL PT TO FN
        MOV     L,E
        MVI     B,11    ; WILD, SO MAKE FN AND FT ALL '?'S
        MVI     A,'?'   ; WILD OVERALL
        CALL    FILL    ; FILL IT
SEARCH$DIR1:
        INX     D       ; PT TO NEXT
        LDAX    D
        CPI     ':'     ; COLON MEANS DRIVE SPEC CAUGHT
        RZ
        LXI     H,DIR1  ; PT TO DIR1 & DIR2
        LXI     D,DIR2
        LDA     FILE$COUNT      ; NUMBER OF ENTRIES IN DIR1
        MOV     B,A     ; ... IN B

*  COMPARE ENTRY IN DIR1 AGAINST FCB
SD1:
        PUSH B ! PUSH D ! PUSH H
        MVI     B,11    ; 11 BYTES IN FN AND FT
        LXI     D,FCB+1 ; PT TO FN IN FCB

*  DISK NAME ALWAYS MATCHES
        MOV     A,M     ; CHECK 1ST BYTE
        CPI     '-'     ; DISK NAME?
        JZ      SD3A    ; MATCH IF SO

*  DO COMPARISON
SD2:
        LDAX    D       ; GET FCB BYTE
        CPI     '?'     ; WILD?
        JZ      SD3
        CMP     M       ; MATCH?
        JNZ     SD4

*  PARTIAL MATCH
SD3:
        INX     H       ; PT TO NEXT
        INX     D
        DCR     B
        JNZ     SD2
```

```
*  CHECK FOR NEGATIVE SELECTION; NO MATCH IF SO
        LDA     NEGFLG  ; NEGATE SELECTION?
        ORA     A       ; 0=NO
        JNZ     SD4A

*  COMPLETE MATCH — COPY INTO DIR2
SD3A:
        POP H ! POP D
        MVI     B,11    ; 11 BYTES/ENTRY
        CALL    MOVE
        JMP     SD5

*  NO MATCH — DECR FILE$COUNT
SD4:
        LDA     NEGFLG  ; NEGATE SELECTION?
        ORA     A       ; 0=NO
        JNZ     SD3A

*  NO MATCH — DECR FILE$COUNT
SD4A:
        LDA     FILE$COUNT
        DCR     A
        STA     FILE$COUNT
        POP     H       ; GET PTR TO DIR1
        LXI     D,11    ; PT TO NEXT ENTRY
        DAD     D
        POP     D       ; GET PTR TO DIR2

*  CONTINUE UNTIL END OF DIR1
SD5:
        POP     B       ; GET CNT
        DCR     B       ; COUNT DOWN
        JNZ     SD1

*  DONE — CHECK IF ANY MATCHES
        LDA     FILE$COUNT
        ORA     A
        RZ

*  AT LEAST ONE MATCH — COPY INTO DIR1 FROM DIR2
        MOV     C,A     ; NUMBER OF ENTRIES
        LXI     H,DIR2  ; FROM DIR2 INTO DIR1
        LXI     D,DIR1
```

```
SD6:
        MVI     B,11    ; 11 BYTES/ENTRY
        CALL    MOVE
        DCR     C       ; COUNT DOWN
        JNZ     SD6
        RET

*
*  PRINT A AS DECIMAL NUMBER 0-255 WITH LEADING <SP>
*
PRINT$VALUE:
        PUSH B ! PUSH D
        MOV     D,A     ; SAVE A
        MVI     A,1     ; SET LEADING <SP> FLAG
        STA     LDSP
        MOV     A,D
        MVI     E,100   ; PRINT 100'S COUNT
        CALL    PDEC
        MVI     E,10    ; PRINT 10'S COUNT
        CALL    PDEC
        ADI     '0'     ; PRINT 1'S COUNT
        CALL    CON$LST$OUT
        POP D ! POP B
        RET
*
*  A=A-E UNTIL <0; PRINT COUNT OF NUMBER OF TIMES THIS OCCURRED
*       USED FOR DECIMAL PRINTING, BUT COULD BE APPLIED TO ANY
*       BASE
*
PDEC:
        MVI     C,0     ; SET COUNT
PDEC1   SUB     E       ; A=A-E
        JC      PDEC2
        INR     C       ; INR CNT
        JMP     PDEC1
PDEC2   ADD     E       ; MAKE A POS
        MOV     D,A     ; SAVE IN D
        MOV     A,C     ; GET CNT
        ORA     A       ; ZERO?
        JNZ     PDEC3
        LDA     LDSP    ; PRINT LEADING <SP>?
        ORA     A
        JZ      PDEC3   ; PRINT AS DIGIT
```

```
        MVI     A,' '   ; PRINT <SP>
        JMP     PDEC4
PDEC3   XRA     A       ; NO LEADING <SP>
        STA     LDSP
        MOV     A,C     ; GET VALUE
        ADI     '0'     ; CONVERT TO ASCII
PDEC4   CALL    CON$LST$OUT
        MOV     A,D     ; RESTORE VALUE
        RET

*
*  PLACE ENTRY INTO DIR1 IF NOT ALREADY THERE; IF ALREADY THERE,
*       THROW AWAY
*
*  ON INPUT,  B=BYTE ADR IN FCB OF ENTRY
*             ENTRY$COUNT=NUMBER OF ENTRIES IN DIR SO FAR
*  ON OUTPUT, ENTRY$COUNT=NUMBER OF ENTRIES IN DIR SO FAR
*
PUT$ENTRY:
        PUSH PSW ! PUSH B
        MVI     E,BUFF  ; ADR IN DE
        MVI     D,0
        MVI     A,4     ; LOOK FOR 4 ENTRIES
        STA     PUT$CNT
        XCHG            ; SAVE PTR TO ENTRY
        SHLD    PUT$LOC

*  PUT AT MOST 4 ENTRIES INTO DIR
PUT$LOOP:
        MOV     A,M     ; ENTRY PRESENT?
        ORA     A
        JNZ     PUT$TEST
PUT1$LOOP:
        XCHG            ; GET PTR BACK
        INX     D       ; PT TO FN
        LXI     H,DIR1  ; PT TO DIR TABLE
        LDA     ENTRY$COUNT     ; EMPTY DIR?
        ORA     A
        JZ      MAKE$NEW$ENTRY
        MOV     C,A     ; NUMBER OF ENTRIES ALREADY IN DIR1

*  LOOK FOR A MATCH
SCAN$DIR:
```

```
        MVI     B,11    ; 11-BYTE MATCH (FN, FT)
        PUSH    H       ; SAVE PTRS
        PUSH    D
SCANL$DIR:
        LDAX    D       ; GET FN OR FT BYTES
        CMP     M       ; COMPARE AGAINST DIR1 BYTE
        JNZ     SCANF$DIR       ; FAIL IF NO MATCH
        INX     H       ; PT TO NEXT BYTE
        INX     D
        DCR     B       ; COUNT DOWN
        JNZ     SCANL$DIR

*  MATCH — REJECT ENTRY AND FALL THRU TO TEST
        POP     D       ; GET PTRS
        POP     H

*  DONE WITH PLACING AT MOST 4 ENTRIES?
PUT$TEST:
        LHLD    PUT$LOC ; GET PTR TO CURRENT ENTRY
        LXI     D,32    ; SKIP TO NEXT
        DAD     D
        SHLD    PUT$LOC
        MOV     A,H     ; BEYOND BUFF?
        CPI     1
        JZ      PUT$DONE
        LDA     PUT$CNT ; DCR COUNT
        DCR     A
        STA     PUT$CNT
        JNZ     PUT$LOOP
PUT$DONE:
        POP B ! POP PSW
        RET

*  NO MATCH
SCANF$DIR:
        POP     D       ; GET PTRS
        POP     H
        PUSH    D       ; SAVE FCB PTR
        LXI     D,11    ; PT TO NEXT ENTRY IN DIR1
        DAD     D
        POP     D       ; RESTORE FCB PTR
        DCR     C       ; DONE W/SCAN?
        JNZ     SCAN$DIR
```

```
*
*  MAKE NEW ENTRY IN DIR TABLE
*       ON INPUT, HL=ADR TO PLACE ENTRY IN DIRl
*                 DE=ADR IN FCB
*
MAKE$NEW$ENTRY:
        MVI     B,11    ; 11 BYTES IN DIR ENTRY
PUT$NAME:
        LDAX    D       ; STORE FN AND FT
        MOV     M,A
        INX     H       ; PT TO NEXT BYTE
        INX     D
        DCR     B
        JNZ     PUT$NAME

*  INCREMENT ENTRY COUNT
        LDA     ENTRY$COUNT
        INR     A
        STA     ENTRY$COUNT
        JMP     PUT$TEST

*
*  GET CHAR FROM CONSOLE
*
CHAR$IN:
        PUSH H ! PUSH D ! PUSH B
        MVI     C,1     ; READ CONSOLE
        CALL    BDOS
        POP B ! POP D ! POP H
        RET
*
*  PRINT CHAR IN A ON CON:
*
CHAR$OUT:
        PUSH H ! PUSH D ! PUSH B ! PUSH PSW
        MOV     E,A     ; CHAR IN E
        MVI     C,2     ; PRINT TO CON:
        CALL    BDOS
        POP PSW ! POP B ! POP D ! POP H
        RET

*
```

```
*  <CR> <LF>
*
CRLF:
        MVI     A,CR    ; PRINT CR
        CALL    CHAR$OUT
        MVI     A,LF    ; PRINT LF
        JMP     CHAR$OUT

*
*  PRINT MESSAGE POINTED TO BY RETURN ADDRESS ON CON:
*       MESSAGE ENDS IN BINARY 0
*
PRINT$MESSAGE:
        XTHL            ; HL ON STACK, HL=PTR TO MSG
PRINT$MESS:
        MOV     A,M     ; GET BYTE
        INX     H       ; PT TO NEXT BYTE
        ORA     A       ; 0=DONE
        JZ      PRINT$DONE
        CALL    CHAR$OUT        ; PRINT
        JMP     PRINT$MESS
PRINT$DONE:
        XTHL            ; RESTORE HL AND RET ADR
        RET

*
*  BASIC SUPPORT ROUTINES
*
ADD$11$TO$HL:
        MVI     A,11    ; 11
ADD$TO$HL:              ; HL=HL+A
        ADD     L       ; ADD A TO HL
        MOV     L,A
        MOV     A,H
        ACI     0
        MOV     H,A
        RET
FILL:                   ; FILL MEMORY PTED TO BY HL WITH CHAR IN A FOR B BYTES
        MOV     M,A     ; PUT CHAR
        INX     H       ; PT TO NEXT
        DCR     B
        JNZ     FILL
        RET
```

```
MOVE:                   ; MOVE DATA PTED TO BY HL TO DE FOR B BYTES
        MOV     A,M     ; GET BYTE
        STAX    D       ; PUT BYTE
        INX     H       ; PT TO NEXT
        INX     D
        DCR     B
        JNZ     MOVE
        RET


*
*  CP/M EQUATES
*
CPM     EQU     0       ; CP/M WARM BOOT
BDOS    EQU     5       ; ENTRY POINT
FCB     EQU     5CH     ; FCB
BUFF    EQU     80H     ; BUFFER
CR      EQU     0DH     ; CR
LF      EQU     0AH     ; LF
CTRLZ   EQU     'Z'-40H ; CTRL-Z

*
*  XDIR EQUATES
*
OPTC            EQU     '/'     ; OPTION INDICATOR CHAR
MAX$CHAR        EQU     80      ; MAX NUMBER OF CHARS IN PBUF


*
*  MESSAGE BUFFERS
*
HELPMS:
        DB      CR,LF
        DB      'CATALOG COMMAND --',CR,LF
        DB      '  CAT x:filename.typ dskname.dsk /o',CR,LF
        DB      ' All arguments are optional, and wild cards (*,?) are',CR,LF
        DB      ' permitted in "filename.typ" and "diskname.dsk".',CR,LF
        DB      'Options are --',CR,LF
        DB      '       /C = invoke CATALOG Function on A:',CR,LF
        DB      '       /E = display CATALOG Environment on CON:',CR,LF
        DB      '       /F <delim> <FILENAME> <delim> = declare',CR,LF
        DB      '            name of CATALOG File ''FILENAME.CAT'';',CR,LF
        DB      '            at most 8 characters may be in FILENAME',CR,LF
```

```
        DB      '       /L = toggle listing of selected files',CR,LF
        DB      '       /N = negate selection',CR,LF
        DB      '            If this option is given, wild card',CR,LF
        DB      '            specifies files NOT to be selected',CR,LF
        DB      '       /S = save CATALOG Environment on disk',CR,LF
        DB      '       /T = toggle default CATALOG/UPDATE Function',CR,LF
        DB      '       /U = invoke UPDATE Function on B: (or x:)',CR,LF
        DB      '$'

PAGEMS:
        DB      'Page Number ',0
STD$HDR:
        DB      CR,LF,LF
        DB      'Filename.Typ   Diskname.Dsk   Diskname.Dsk   Diskname.Dsk'
        DB      '   Diskname.Dsk',CR,LF,LF,0

*
*  BUFFERS
*
CAT$LOAD$FLAG:
        DS      1       ; 0=>NO ENTRY IN WORK$FILE/WORK$DISK; 1=>ENTRY
PAGE$NUM:
        DS      1       ; PAGE NUMBER
PAGE$COUNT:
        DS      1       ; PAGING COUNT
PRINT$FLAG:
        DS      1       ; PRINT FLAG
FILE1$BUF:
        DS      11      ; 11 BYTES FOR FILE NAME AND TYPE
LAST$FILE:
DISK$NAME:
        DS      11      ; NAME OF CUR DISK FOR UPDATE/NAME OF PREV FILE FOR CAT
WORK$FILE:
        DS      11      ; NAME OF WORKING FILE FROM DISK
WORK$DISK:
        DS      11      ; NAME OF WORKING DISK FROM DISK
DIR$PTR:
        DS      2       ; PTR TO DIR1 ENTRY
LOG$DRIVE:
        DS      1       ; DRIVE NUMBER TO LOG IN FOR CATALOG
CHNGFLG:
        DS      1       ; CHANGE FLAG; 0=NO CHANGE ON DISK, 0FFH=CHANGE
PUT$CNT:
```

```
        DS      1       ; PUT$ENTRY BUFFER FOR FILE COUNT
PUT$LOC:
        DS      2       ; POSITION OF CURRENT DIRECTORY ENTRY IN BUFF
NEGFLG:
        DS      1       ; NEGATE SELECTION FLAG; 0=NO NEGATION, 0FFH=NEGATE
FILE$COUNT:
        DS      1       ; NUMBER OF FILES IN DIR
ENTRY$COUNT:
        DS      1       ; NUMBER OF ENTRIES IN DIR
PBUF:
        DS      MAX$CHAR+1      ; PRINT HEADING BUFFER
TFCB:
        DS      33      ; TEMP FCB
TITLE$LINE:
        DB      ' --   ',0      ; PAGE NUMBER/TITLE SEPARATOR
TBUFF:
        DS      128     ; TEMP BUFFER AREA
LDSP:
        DS      1       ; LEADING <SP> FLAG
LFLG:
        DS      1       ; LOGIN FLAG; 0FFH=NO CHANGE, OTHERWISE DRIVE
        DS      40      ; XDIR STACK SPACE
STACK   EQU     $
WORK$BYTE:
        DS      1       ; BUFFER FOR GET/PUT
GET$BUFFER:
        DS      128     ; BUFFER FOR GET
GET$BUFFER$PTR:
        DS      2       ; PTR FOR GET
GET$BYTE$CNT:
        DS      1       ; BYTE COUNT FOR GET
PUT$BUFFER:
        DS      128     ; BUFFER FOR PUT
PUT$BUFFER$PTR:
        DS      2       ; PTR FOR PUT
PUT$BYTE$CNT:
        DS      1       ; BYTE COUNT FOR PUT
READ$FILE$PTR:
        DS      2       ; PTR TO FILE FCB FOR GET
WRITE$FILE$PTR:
        DS      2       ; PTR TO FILE FCB FOR PUT
DIR1:
        DS      11*64+1 ; 11 BYTES/ENTRY; 64 ENTRIES MAX; END IN CTRL-Z
```

```
DIR2:
        DS      11*64   ; SAME FOR DIR2

        END
```